UNIVERSIDADE FEDERAL DE SANTA CATARINA

CENTRO TECNOLÓGICO

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Aplicação Web para Controle de Estoques do Laboratório de Pesquisas Toxicológicas da Universidade Federal de Santa Catarina

Camila Maia Cardozo

Florianópolis, dezembro de 2016

UNIVERSIDADE FEDERAL DE SANTA CATARINA

CENTRO TECNOLÓGICO

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Aplicação Web para Controle de Estoques do Laboratório de Pesquisas Toxicológicas da Universidade Federal de Santa Catarina

Camila Maia Cardozo

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

Florianópolis, dezembro de 2016

Camila Maia Cardozo

Aplicação Web para Controle de Estoques do Laboratório de Pesquisas Toxicológicas da Universidade Federal de Santa Catarina

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Leandro José Komosinski

Banca examinadora

Prof. Dr. Frank Augusto Siqueira

Prof. Dr. Ricardo Pereira e Silva

Agradecimentos

À Universidade Federal de Santa Catarina, seu corpo docente, direção e administração, por me proporcionarem a oportunidade de aprendizado durante todo o curso de Sistemas de Informação.

Ao meu orientador, Prof. Dr. Leandro José Komosinski, pela orientação, paciência e por todo conhecimento passado ao longo da graduação.

Aos professores Ricardo Pereira e Silva e Frank Augusto Siqueira, por aceitarem participar da banca do trabalho e pelas contribuições fornecidas.

Aos profissionais da equipe do LPTox, em especial Alcíbia Helena de Azevedo Maia, Maitê Perin e Ana Caroline Hillesheim da Cruz, os quais cederam seu tempo e dedicação para criação deste projeto

À minha família, em especial à minha mãe e ao meu irmão, por me apoiarem e incentivarem sempre.

Ao meu amor, Alexandra Monteiro Stenstrasser, pela suporte e dedicação diária, além da participação direta na criação da visão do sistema.

Ao Paulo Henrique Cardoso, pelo suporte na implantação do sistema no ambiente de produção da UFSC.

À Katiany Zimmermann e Vanessa da Conceição pelos auxílios na montagem do presente documento.

A todos os meus amigos, que me fazem uma pessoa muito mais feliz e realizada.

A todos que, de alguma forma, contribuíram com meu trabalho e minha formação.

Resumo

Gerenciar estoques é um problema conhecido há bastante tempo na administração. Garantir que os produtos necessários estejam à disposição e evitar perdas é uma balança difícil de equilibrar. Laboratório de Pesquisas Toxicológicas (LPTox) da Universidade Federal de Santa Catarina (UFSC) enfrenta esta questão e não tem conseguido solucioná-la com eficácia e eficiência até hoje. Materiais acabam, produtos estão fora da validade e nada disso é percebido até que alguém se depare com o problema, causando perda de tempo e dinheiro.

Desta forma, o presente trabalho apresenta como solução uma aplicação web que auxilia no controle de estoque do LPTox. A ferramenta otimiza a gerência de estoque, além do registro dos itens armazenados de maneira clara e organizada, também emite alertas de quando materiais estão por acabar ou estão prestes a sair da validade; permite buscas de materiais por meio de filtros, possibilitando visões diferentes do todo; e gera documentos que facilitam o pedido de novos materiais.

O protótipo encontra-se ainda em fase de desenvolvimento. Os usuários afirmam que, com as funcionalidades atuais que o *software* apresenta, tem-se 70% dos casos de uso cobertos. Com a implementação do filtro de materiais, esse número já atingiria a casa dos 90%.

Palavras-chave: Gerenciamento de estoque, estoque, laboratório, Laboratório de Pesquisas Toxicológicas, aplicação web.

Abstract

Manage inventory is a known issue for a long time. It is really difficult to ensure that products are available and to avoid waste. Laboratory of Toxicological Research (LPTox) of the Federal University of Santa Catarina (UFSC) faces this issue and has not been able to solve it effectively and efficiently nowadays. Lack of materials, products out of date and no one notice the problem, causing loss of time and money.

Thus, the present study proposes a web application to assist the LPTox/UFSC inventory control. The software optimizes inventory management, in addition to the record the stored items in a clear and organized way, also sends alerts when supplies are finishing or are about to be out of shelf life; it allows searches of materials through filters, allowing different views of the whole picture; and generates documents that facilitate the order of new materials.

The prototype is still in the development phase. Until now, the users claim the inventory management software covers about 70% of the use cases. With the implementation of filter materials feature, this number would reach around 90%.

Keywords: Inventory management, inventory, laboratory, Toxicological Research Laboratory, web application.

Lista de Tabelas

Lista de Ilustrações

Lista de Abreviaturas

UFSC	Universidade Federal de Santa Catarina
LPTox	Laboratório de Pesquisas Toxicológicas
CCS	Centro de Ciências da Saúde
PTL	Patologia
MVC	Padrão model-view-controller
HTTP	HyperText Transfer Protocol
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
EAV	Entity–attribute–value model

Sumário

1 Introdução

Laboratórios de química em geral possuem muitos reagentes, vidrarias, utensílios e outros itens que precisam ser armazenados e controlados apropriadamente. O controle de estoque é uma área da administração que consiste em gerenciar tudo o que envolve cada item de um estoque, desde fornecedores, pedidos, localização dos itens, quantidades, validades e o quê mais for necessário para o controle pleno desta área.

Entretanto, gerenciar um estoque nem sempre é uma tarefa fácil. Quando não bem executada, pode causar sérios prejuízos ao laboratório, como perda de dinheiro, ausência de materiais para usos primordiais, materiais fora da validade e claro, muita perda de tempo ao ter que trabalhar com este caos.

O Laboratório de Pesquisas Toxicológicas (LPTox) da Universidade Federal de Santa Catarina (UFSC) possui um vasto estoque e se enquadra nesta situação caótica citada anteriormente. Atualmente, o laboratório tenta desenvolver seu próprio método de controle de estoque, porém, de maneira falha e ineficiente. Materiais acabam, produtos estão fora da validade e nada disso é percebido até se deparar com o problema. Aliado a isso, existe ainda a grande perda de tempo no processo de novos pedidos de materiais.

O gerenciamento de estoques de laboratório é notoriamente imprescindível para um bom funcionamento do laboratório. Pode-se avaliar a grandiosidade do problema analisando os softwares já existentes no mercado. A aplicação Quartzy, por exemplo, é utilizado por mais de 30.000 laboratórios ao redor do mundo (KULKARNI et al., 2016), reforçando que o problema é sim real. Contudo, as necessidades do LPTox são um pouco mais específicas e os programas existentes no mercado analisados não eram totalmente adequados para a tarefa. Idioma em Português, possibilitar o gerenciamento de mais de um laboratório com a mesma conta e emitir modelos de pedidos de novos materiais no padrão UFSC são exemplos de funcionalidades que não foram encontradas em outros sistemas, mas que seriam importantes para o LPTox. Desta forma, a proposta deste *software* vem para contemplar as necessidades de gerenciamento de estoque do laboratório em questão.

A metodologia utilizada para o desenvolvimento da aplicação foi baseada no *framework* Scrum, visando um desenvolvimento ágil e eficaz. O protótipo se

encontra ainda em fase de desenvolvimento. Em entrevista com a supervisora do LPTox, ela afirma que, com as funcionalidades atuais que o *software* apresenta, tem-se 70% dos casos de uso cobertos. Com a implementação do filtro de materiais, esse número já atingiria a casa dos 90%.

Este trabalho de conclusão de curso se estrutura em seis capítulos. O primeiro capítulo introduz ao leitor o contexto do projeto, apresentando o LPTox e o cenário atual do gerenciamento de estoque nele, bem como os objetivos deste estudo. O segundo capítulo é responsável por fornecer a fundamentação técnica para o entendimento de todo o trabalho. O terceiro capítulo mostra a pesquisa realizada a respeito das soluções já existentes de gerenciamento de estoque de laboratórios. No quarto capítulo, é apresentada a solução produzida, quais os requisitos da aplicação e qual arquitetura e metodologia foram adotadas. O quinto capítulo caracteriza a implementação da aplicação, introduzindo as tecnologias utilizadas, o processo de levantamento de requisitos e relatando os resultados de cada etapa de desenvolvimento. O sexto e último capítulo traz as considerações finais do trabalho, quais conclusão foram tiradas e propostas para trabalhos futuros.

1.1 Laboratório de Pesquisas Toxicológicas

O LPTox do Centro de Ciências da Saúde (CCS) está vinculado ao departamento de Patologia (PTL) da UFSC. Até o início do ano de 2003, o Laboratório de Toxicologia funcionava na rua Ferreira Lima, no centro de Florianópolis, nas dependências da antiga Faculdade de Medicina, quando foi transferido para o bloco K do CCS, salas 307, 308, 309 e 310.

Atualmente, o referido laboratório desempenha atividades de ensino com aulas práticas semanais de toxicologia para o Curso de Farmácia abrangendo principalmente as áreas de toxicologia de medicamentos, toxicologia social e toxicologia ocupacional. As análises são realizadas em material não biológico e biológico.

Na pesquisa atua no campo da toxicologia forense, através da identificação de drogas de abuso em amostras biológicas alternativas, e na toxicologia ambiental e ocupacional, com quantificação de metais em amostras biológicas.

Em relação às atividades de extensão, são desenvolvidos no laboratório, por exemplo, projetos voltados para alunos de graduação da UFSC, como a organização

de cursos em Ciências Forenses, por meio de uma parceria realizada com o Instituto Geral de Perícias do Estado de Santa Catarina. Outro exemplo deste tipo de atividade são os cursos disponibilizados para professores e alunos do ensino médio visando apoio à melhoria do ensino de ciências nas escolas públicas da Grande Florianópolis.

O laboratório conta com uma equipe de profissionais graduados em Farmácia e Química, dentre eles professores doutores e técnica de laboratório, que desempenham suas atividades voltadas para alunos de graduação e pós-graduação.

Para desenvolvimento de suas atividades o LPTox possui os seguintes equipamentos, entre outros:

- Cromatógrafo Líquido de Alta Eficiência com arranjo de diodos;
- Espectrofotômetro de Absorção Atômica com forno de grafite;
- Equipamento para geração de Vapor a frio;
- Cromatógrafo Gasoso com detector por ionização de chamas;
- Espectrofotômetro UV/Visível;
- Concentrador de amostras;
- Sonicador;
- Capelas de fluxo laminar e de exaustão;
- Centrífugas.

Neste contexto, considerando as diferentes atividades exercidas pelo LPTox, é consenso entre os profissionais do laboratório, quanto à necessidade de se ter algum tipo de controle sobre os inúmeros itens utilizados, dentre estes, reagentes, utensílios, vidrarias, para que a rotina do laboratório não seja prejudicada.

1.2 Cenário Atual

Atualmente o gerenciamento dos materiais é feito via tabelas no Excel, *software* da Microsoft especializado em planilhas. A responsável pela atualização dos dados no programa é a técnica do laboratório.

Os materiais são organizados em três estoques diferentes: estoque geral, estoque para pesquisa e estoque para ensino. Os três estoques se encontram, inclusive, em espaços físicos diferentes. A responsável pelo estoque geral é também a técnica do laboratório. Os responsáveis pelo estoque de pesquisa são os alunos

que estão fazendo algum tipo de pesquisa e o responsável pelo estoque de aula é o monitor da disciplina de Toxicologia.

A dinâmica consiste no seguinte: a maior parte dos materiais se encontra no estoque geral, porém, quando o monitor ou algum pesquisador precisa de materiais, eles pedem à técnica do laboratório, que retira o que foi pedido do estoque geral e passa para o respectivo estoque destino - ensino ou pesquisa. À medida que alterações deste tipo são feitas, assim como qualquer outra alteração no estoque geral, a técnica altera as planilhas no mesmo instante.

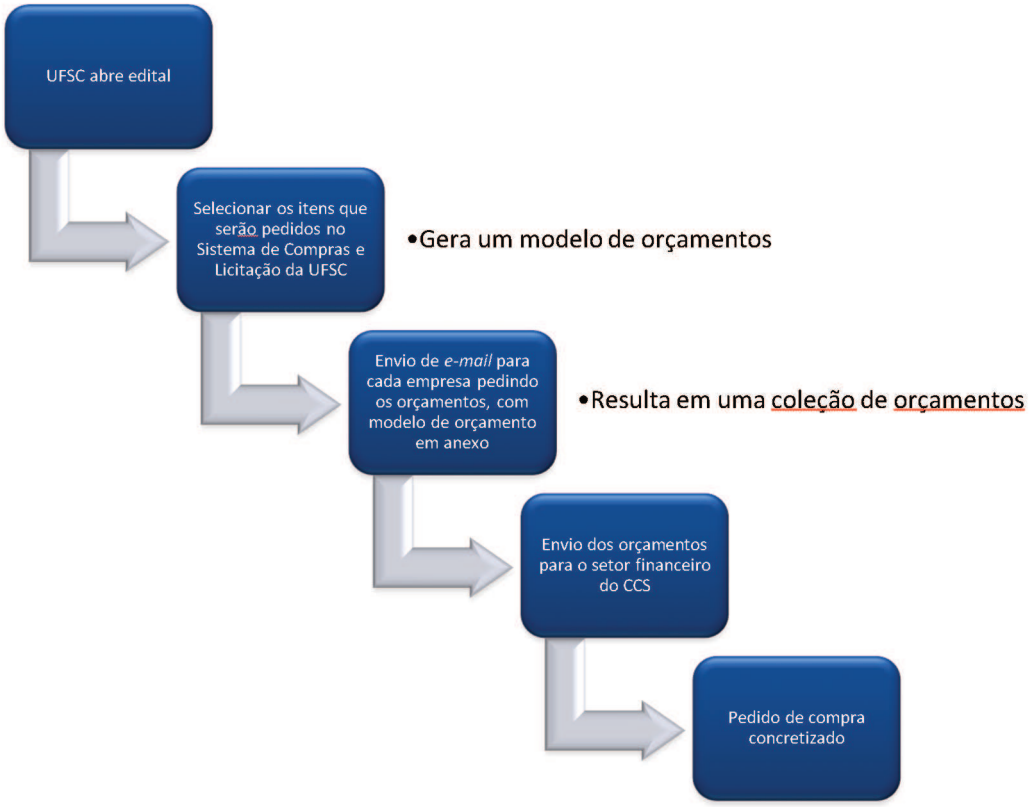
Algumas dificuldades são encontradas nessa dinâmica. Saber a quantidade de reagente em cada estoque, é um exemplo. Nem sempre o/a monitor(a) ou os que estão realizando pesquisa informam precisamente a técnica de quanto material foi utilizado. Às vezes, informam apenas quando o material já acabou, forçando a supervisora a ter que conseguir o material às pressas, caso não tenha também no estoque geral. Outro exemplo de dificuldade é controlar materiais que já foram parcialmente utilizados. A embalagem nem sempre está cheia e um reagente pode já ter sido utilizado parcialmente, mas essa informação não consta na planilha. Existem mais dois problemas recorrentes no LPTox: saber quando um material saiu da validade e saber em que estado de conservação o material estocado se encontra através das planilhas. Para pesquisas, principalmente, é fundamental saber se o material está dentro da validade, para garantir a relevância dos experimentos. Muitas vezes precisa-se de um material, que até se encontra no estoque, porém fora da validade, impossibilitando a sua utilização. O mesmo ocorre para o estado de conservação, há situações em que o material se encontra no estoque, porém em um mal estado de conservação, impedindo seu uso.

Além do armazenamento, é preciso compreender o processo de aquisição de materiais. No referido contexto, existem três tipos de compras que podem ser realizadas: via UFSC, por meio de licitação; via CCS, por meio de memorando; compra direta, por meio do CPF da supervisora.

A primeira opção é a que mais toma tempo e trabalho da técnica de laboratório. Quando a UFSC abre um edital para compra de materiais, a técnica precisa obter orçamento de 3 empresas diferentes para cada item a ser comprado; em que os valores dos 3 orçamentos precisam ser parecidos, já incluindo o frete, e que os orçamentos possuam validade de pelo menos 10 dias. Primeiramente a técnica entra em contato com o setor financeiro do CCS, para selecionar os itens que ela

deseja pedir no Sistema de Compras e Licitação da UFSC. O sistema gera um modelo de orçamento, conforme a Figura 1.2, contendo todos os itens com seus respectivos campos de preços que devem ser preenchidos pelas empresas. Então, ela envia um *e-mail* para cada empresa pedindo os orçamentos, com este modelo em anexo. Como as empresas não possuem sempre todos os itens da lista pedida, são necessários vários *e-mails* para várias empresas. Após conseguir todos os orçamentos, eles são enviados para o setor financeiro do CCS, concretizando o pedido de compra. Este processo é ilustrado na Figura 1.1.

Figura 1.1 – Processo de pedido de compra via UFSC.



Fonte: autora.

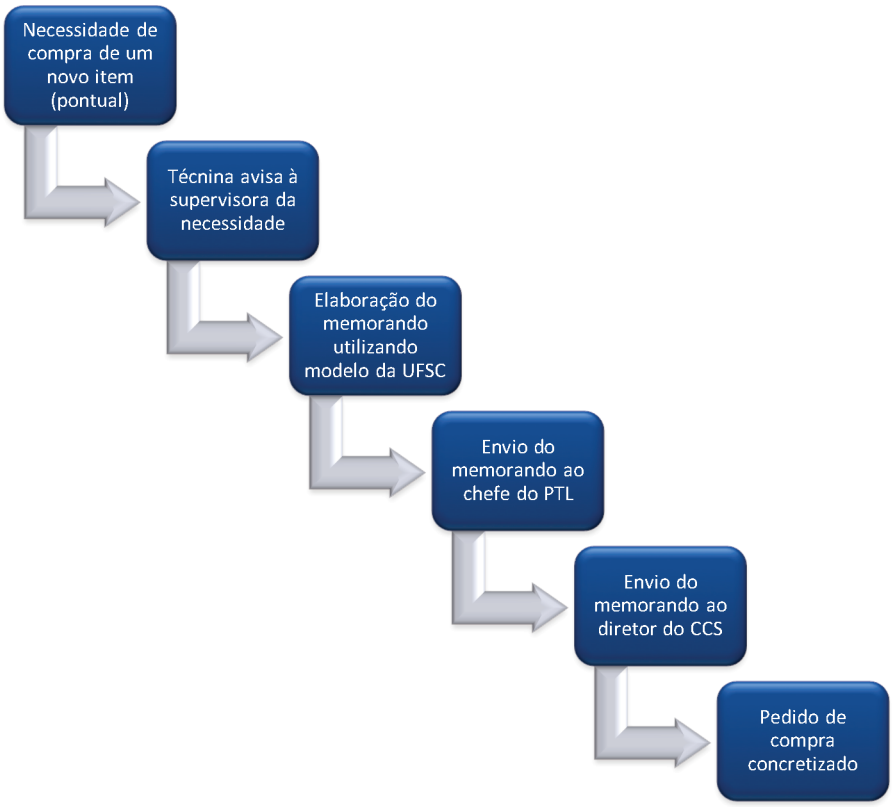
Figura 1.2 – Modelo de orçamento gerado pelo Sistema de Compras e Licitação da UFSC.

Nº	Item	Unid.	Qtde	Especificação	Valor unitário	Marca/Modelo	Total (R$)
099.03.070712	BECKER - FORMA ALTA -	UN	127,0000	Ler			0,00
099.03.070715	BECKER - FORMA ALTA -	UN	67,0000	Ler			0,00
099.03.070719	BECKER - FORMA ALTA -	UN	127,0000	Ler			0,00
099.03.070720	BECKER - FORMA ALTA -	UN	127,0000	Ler			0,00
099.03.070722	BECKER - FORMA ALTA -	UN	127,0000	Ler			0,00
099.03.290340	LAMÍNULA - RETANGULAR -	CX	12,0000	Ler			0,00
099.03.290351	PROVETA - VIDRO	UN	28,0000	Ler			0,00
099.03.290504	PROVETA - VIDRO	UN	136,0000	Ler			0,00
099.03.290582	PROVETA - VIDRO	UN	137,0000	Ler			0,00
099.03.290585	ERLENMEYER BOCA ESTREITA	UN	67,0000	Ler			0,00
099.03.290586	ERLENMEYER BOCA ESTREITA	UN	67,0000	Ler			0,00
099.03.290592	PROVETA - VIDRO	UN	132,0000	Ler			0,00
099.03.290592	PROVETA - VIDRO	UN	65,0000	Ler			0,00

Fonte: autora.

A segunda categoria de pedidos de compra é via CCS. Compras deste tipo são caracterizadas por serem solicitadas fora dos editais, sendo necessárias ocasionalmente, são mais pontuais. Quando vidrarias são quebradas, por exemplo, a técnica avisa à supervisora de laboratório do ocorrido, que faz um memorando justificando a necessidade da compra dos novos itens, utilizando um modelo da UFSC. O memorando é enviado ao chefe da PTL e depois para o diretor do CCS, finalizando o pedido.

Figura 1.3 – Processo de pedido de compra via CCS.



Fonte: autora.

O terceiro, e último, processo de compra é a compra direta. Neste, as compras são feitas por meio de uma pessoa física, no caso, a supervisora de laboratório. Quando materiais precisam ser comprados, a técnica pede orçamentos para as empresas e os envia para a supervisora. A supervisora analisa quais deseja comprar, efetuando a compra utilizando o seu CPF e pagando com o dinheiro do laboratório adquirido por meio de projetos.

Para os três tipos de pedido de compras existe um mesmo problema: saber exatamente quais itens pedir. Saber quais itens são necessários para agora ou para um futuro próximo, se estão em falta, fora da validade ou ainda mal conservados. Atualmente a técnica analisa as planilhas que possui e tenta encontrar o quê deve ser pedido, mas este processo não tem sido eficaz e nem eficiente.

Além disso, o pedido via UFSC, enfrenta um grande empecilho que é a demora para chegada dos materiais, bem como o trabalho para enviar os orçamentos para todas as empresas até conseguir os orçamentos necessários. Via compra direta, não se tem acesso ao Sistema de Compras e Licitação da UFSC para selecionar

diretamente os itens desejados e gerar um modelo de orçamento, então gasta-se muito tempo copiando os itens das planilhas a fim de enviá-los para cada empresa pedindo orçamento.

O sistema aqui proposto apresenta-se para solucionar os principais problemas enfrentados neste cenário atual, como veremos na seção seguinte.

1.3 Objetivos

O objetivo deste trabalho é desenvolver uma aplicação web que auxilie no gerenciamento dos estoques do LPTox da UFSC. Que além disso, através do sistema desenvolvido, os colaboradores do referido laboratório possam gerar documentos que relatem o estado atual dos estoques através da aplicação de filtros de consulta; que estes usuários sejam alertados quando materiais estão quase acabando ou quase saindo da validade; e que os mesmos possam gerar modelos para pedidos de compras de novos materiais. O intuito é que o LPTox invista menos tempo com a manutenção do estoque em si, porém tendo a garantia de que os materiais estão bem monitorados.

1.3.1 Objetivo Geral

Desenvolver uma aplicação web que auxilie no gerenciamento dos estoques do Laboratório de Pesquisas Toxicológicas da Universidade Federal de Santa Catarina.

1.3.2 Objetivos Específicos

- Gerar documentos que relatem o estado atual dos estoques através da aplicação de filtros de consulta.
- Gerar um modelo de pedido de orçamento a partir de materiais previamente selecionados dos estoques.
- Alertar aos usuários do sistema quando materiais estão quase acabando ou quase saindo da validade.
- Gerar um modelo de memorando de pedido de compras a partir de materiais previamente selecionados dos estoques.

2 Fundamentação Teórica

2.1 Estratégias de desenvolvimento

2.1.1 Scrum

Scrum é um *framework* de desenvolvimento ágil de *software* iterativo e incremental para gerenciar o desenvolvimento de produtos (SCRUM, 2016) (VERHEYEN, 2013). Ainda, segundo o guia “The Scrum Guide” (SUTHERLAND; SCHWABER, 2016), é “Um *framework* no qual as pessoas podem resolver problemas complexos adaptativos, enquanto produtiva e criativamente entregam produtos de maior valor possível.” Em ambos os casos, entende-se *framework* como uma caixa de ferramentas, que neste contexto, visa o aperfeiçoamento do gerenciamento de produtos.

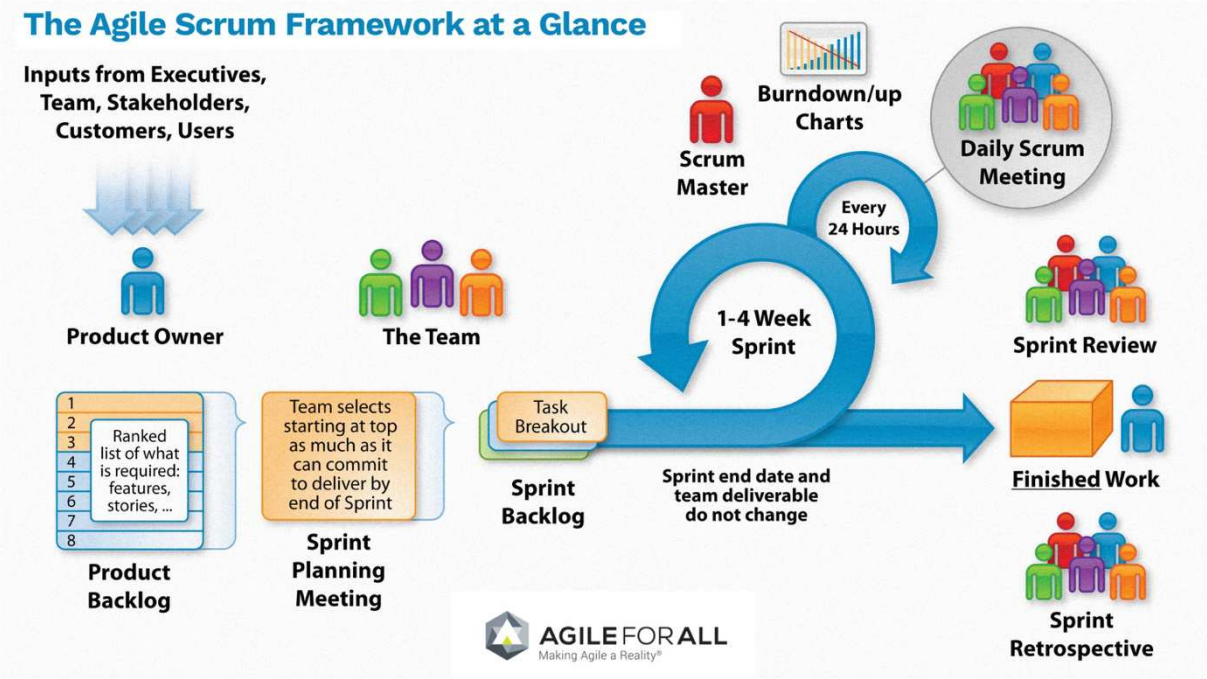
O Scrum consiste em times (*Scrum Teams*), seus membros, eventos, artefatos e regras. Cada time é composto por um *Product Owner*, responsável por maximizar o valor do produto e do trabalho do time de desenvolvimento; por um time de desenvolvimento (*Development Team*), responsável por realizar as tarefas do *Sprint*; e por um *Scrum Master*, que tem como papel garantir que o Scrum seja compreendido e seguido. Deve-se ressaltar que o time de desenvolvimento deve ser totalmente auto gerenciável, pequeno - deve conter cerca de três a nove membros - e de preferência deve ser alocado em um ambiente protegido de distrações externas.

Uma vez que o Scrum é iterativo, existem eventos que ocorrem periodicamente, com o intuito de criar regularidade e minimizar a necessidade de reuniões não previstas pelo Scrum. São eles: *Sprint,* a unidade básica do desenvolvimento, deve ser um esforço restrito a uma duração específica, um mês ou menos; *Sprint Planning*, o evento em que o *Sprint* deve ser planejada; *Daily Scrum*, uma reunião diária de 15 minutos para a equipe de desenvolvimento sincronizar as atividades e criar um plano para as próximas 24 horas; *Sprint Review*, reunião realizada no final do *Sprint* para avaliar e adaptar o *Product Backlog* se necessário; *Sprint Retrospective*, é uma oportunidade para o time se avaliar e criar um plano de melhorias a ser executado durante o próximo *Sprint*.

Além dos membros e eventos, parte importante do Scrum também são seus artefatos. Os artefatos são: *Product Backlog*, uma lista ordenada de tudo o que é necessário no produto (é a única fonte de requisitos para todas as mudanças a serem feitas no produto); *Sprint Backlog*, um subconjunto de itens do *Product Backlog* selecionado para serem executados no *Sprint*.

Conhecendo os membros, eventos e artefatos que compõem o Scrum, falta compreender o fluxo deste *framework*. Inicialmente, por meio de insumos vindos de diversas fontes, o *Product Owner* desenvolve o *Product Backlog*. A partir deste ponto, cada *Sprint* começa com um *Sprint Planning*, visando definir o *Sprint Backlog*. Durante o *Sprint*, todo o dia ocorre a *Daily Scrum* e no final de cada *Sprint* realiza-se a *Sprint Review* e a *Sprint Retrospective.* Este fluxo pode ser visualizado e melhor compreendido na figura 2.1, em que cada etapa é apresentada, mostrando os membros, artefatos e eventos envolvidos.

Figura 2.1 – Visão geral do Scrum



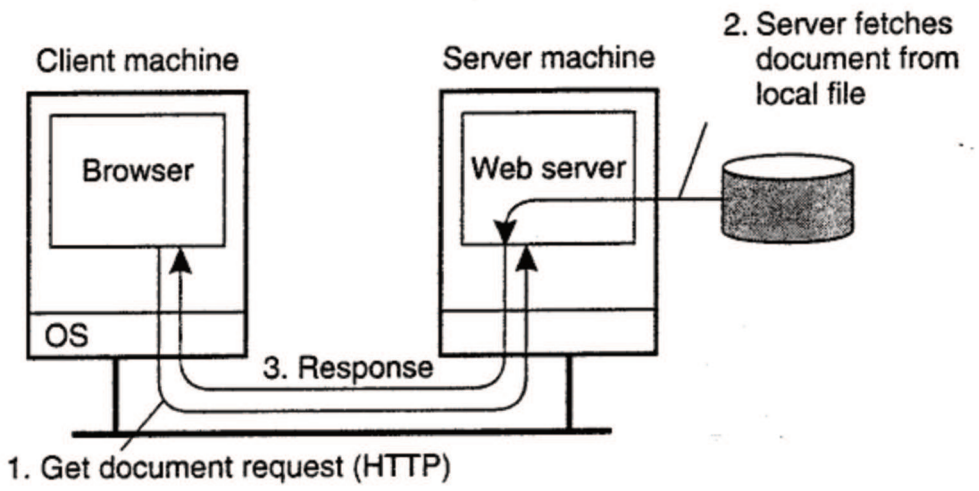
Fonte: http://agileforall.com/resources/introduction-to-agile/

2.2 Aplicação Web

Para compreender o quê é uma aplicação web, inicialmente precisa-se esclarecer o significado de web. Segundo Tanenbaum A. e Steen M no livro Distributed Systems: Principles and Paradigms (TANENBAUM; VAN STEEN, 2006), “A *World Wide Web* (WWW) - ou apenas web - pode ser vista como um grande sistema distribuído que consiste de milhões de clientes e servidores para acessar documentos por meio de *links*”. Seguindo esta ideia, os servidores web possuem a responsabilidade de guardar os documentos, enquanto o cliente tem como objetivo fornecer aos usuários uma interface gráfica de fácil utilização para possibilitar a apresentação e o acesso a tais documentos. Vale ressaltar que a ideia inicial de documentos evoluiu muito desde a criação da web em 1990. Documentos deixaram de ser puramente estáticos e passivos para serem gerados dinamicamente contendo todos os tipos de elementos ativos, além do suporte a serviços (TANENBAUM; VAN STEEN, 2006).

Entendendo-se o conceito de web, define-se aplicação web, ou “web app”, como um *software* que é executado em um servidor web (TECHTERMS, 2016). A arquitetura de sistemas distribuídos para web não é fundamentalmente diferente de outros sistemas distribuídos. O cliente, neste caso, é o navegador, um programa especializado em mostrar apropriadamente documentos, que permite também, a interação com o usuário. O servidor, como já mencionado anteriormente, é o servidor web, máquina responsável por executar a aplicação web. A comunicação entre cliente e servidor, neste cenário, ocorre sempre via protocolo *HyperText Transfer Protocol* (HTTP). A figura 2.2 ilustra a arquitetura tradicional para aplicações web.

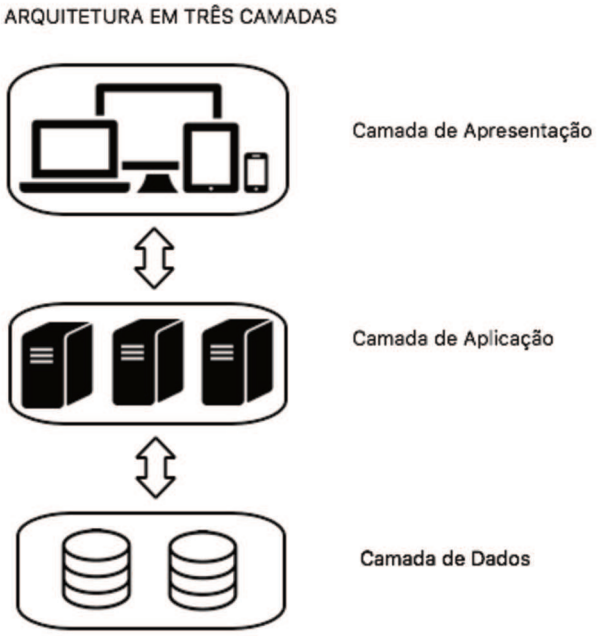
Figura 2.2 – Arquitetura tradicional de uma aplicação web



TANENBAUM, Andrew S.; VAN STEEN, Maarten. **Distributed Systems: Principles and Paradigms.** 2. ed. Upper Saddle River: Pearson Prentice Hall, 2006.

A implementação desta arquitetura no projeto em questão foi realizada utilizando três camadas: Camada de Apresentação, Camada de Aplicação e Camada de Dados, conforme a figura 2.3.

Figura 2.3 – Arquitetura em três camadas.

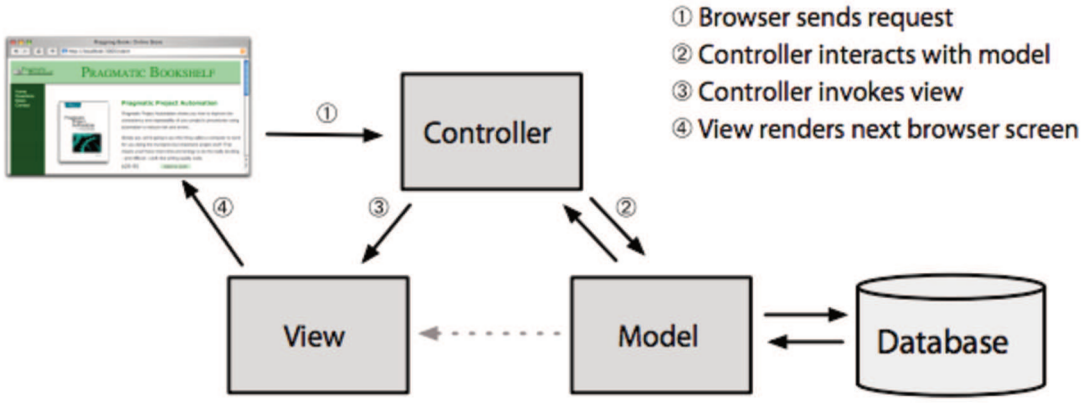


Fonte: autora.

A camada de apresentação é a que se encontra no topo das camadas. É executada no computador do usuário e é responsável por exibir e coletar dados enviados por ele. Para o caso desta aplicação desenvolvida, existe pouco processamento nesta camada. A segunda camada é a Camada de Aplicação. É executada no servidor web e contém a lógica referente à interface gráfica e ao domínio do problema. É onde se encontra de fato o código da aplicação. A terceira e última camada é a Camada de Dados. Para este caso, esta camada se encontra no mesmo servidor que hospeda a aplicação. Sua função é basicamente armazenar os dados do sistema.

Para a implementação da Camada de Aplicação foi utilizado o padrão de arquitetura de *software* model-view-controller (MVC), em português modelo-visão-controlador. O MVC separa a aplicação em três tipos de componentes: modelo, visão e controlador. O modelo é responsável por manter o estado da aplicação, que no caso de ser permanente é comumente armazenado em um banco de dados. Um modelo é mais do que apenas dados; ele impõe todas as regras de negócio que se aplicam a esses dados. A visão é responsável por gerar uma interface de usuário, normalmente baseada em dados do modelo. Os controladores coordenam a aplicação, recebem eventos do navegador, no caso da web, interagem com o modelo e exibem a visão apropriada para o usuário (RUBY et al., 2013). A figura 2.4 ilustra o padrão de arquitetura de *software* MVC.

Figura 2.4 – Padrão de arquitetura de *software* MVC.



Fonte: RUBY, Sam; THOMAS, Dave; HANSSON, David Heinemeier. **Agile Web Development with Rails 4.** 4. ed. Dalas: Rails, 2013.

Com a implementação do MVC visou-se obter a separação da lógica e da apresentação do programa, permitindo a manutenção isolada de ambas as partes. Sendo assim, consegue-se efetivamente maior controle sobre a aplicação,

Portanto, a aplicação desenvolvida é uma aplicação web, um sistema distribuído implementado em três camadas, na qual adotou-se o padrão MVC para a Camada de Aplicação.

3 Soluções existentes

Em busca de conhecer mais sobre a área, de entender a importância do projeto e de aprimorar a aplicação proposta, realizou-se uma pesquisa sobre as soluções já existentes de gerenciamento de estoque de laboratórios. Como não foi previsto nenhum tipo de investimento, foram pesquisadas apenas aplicações gratuitas. Outro ponto a se levar em consideração é o interesse por apenas programas de gerenciamento de estoques voltados especificamente para laboratórios. Espera-se que o *software* seja voltado a esse contexto.

Dentre os requisitos acima, foram encontrados *softwares* que seguem o mesmo padrão: foram desenvolvidos por estudantes para solucionar o problema nos laboratórios de suas universidades ou instituto. Dentre as soluções, o primeiro grupo a ser tratado são as soluções brasileiras: Sistema de Gestão de Reagentes em Laboratórios Universitários - Universidade Federal de São Carlos (CARVALHO et al., 2010); Sistema de Controle de Reagentes – Universidade Federal do Rio Grande (NÚCLEO DE TECNOLOGIA DA INFORMAÇÃO, 2016) e LabStocker – Instituto Federal do Rio Grande do Norte (LIMA, 2015). O primeiro possui foco na redução do desperdício de reagentes e da sua compra desnecessária, visando minimizar a geração de resíduos e o impacto ambiental (CARVALHO et al., 2010). O segundo propõe a gestão do estoque de reagentes químicos das unidades acadêmicas e de seus laboratórios. O último, e aparentemente mais completo, é uma plataforma web de gestão de reagentes que tem como meta melhorar a produtividade da pesquisa em laboratórios de química, por meio de recursos inteligentes de controle de estoque (LIMA, 2015), além de ser um sistema em nuvem que pode ser utilizado de diferentes dispositivos simultaneamente.

Sobre as três soluções, dois pontos foram críticos para a sua não utilização: são soluções específicas para reagentes, não abrangendo utensílios, vidrarias e padrões; não foi possível ter acesso às aplicações – apesar de existirem artigos as citando, não foi encontrado nenhum *link* que levasse ao acesso da aplicação. Uma causa possível seria por serem restritas ao contexto de suas respectivas universidades ou instituto. A aplicação LabStocker até apresenta um site, http://labstocker.com/, porém só consta atualmente um *link* para contato.

No contexto das aplicações criadas fora do país, a aplicação Quartzy aparece como referência. Quartzy é o software número 1 em gerenciamento de laboratório

do mundo, usado por mais de 30.000 laboratórios em instituições acadêmicas, assim como em empresas farmacêuticas e de biotecnologia (KULKARNI et al., 2016). O sistema se apresenta bem completo e funcional. Contudo, as necessidades do LPTox são mais específicas e o *software* não se adequa totalmente às necessidades. Idioma em Português, possibilitar o gerenciamento de mais de um laboratório com a mesma conta e emitir modelos de pedidos de novos materiais no padrão UFSC são exemplos de funcionalidades que não são apresentadas, mas que são importantes para o laboratório. Ainda assim, conhecer o Quartzy também proporcionou insumos para novas funcionalidades. Por exemplo, a importação de dados no formato xls mostra-se bem importante e poderia, constar em uma versão futura deste projeto. Além do mais, mostra também a importância do problema, e de como as soluções aqui implantadas estão no caminho certo, mesmo que em uma equipe muito menor de desenvolvedores, no caso apenas uma, conceitualmente as soluções se assemelham.

4 Solução proposta

4.1 Metodologia

4.1.1 Adaptação do Scrum para Um Único Membro

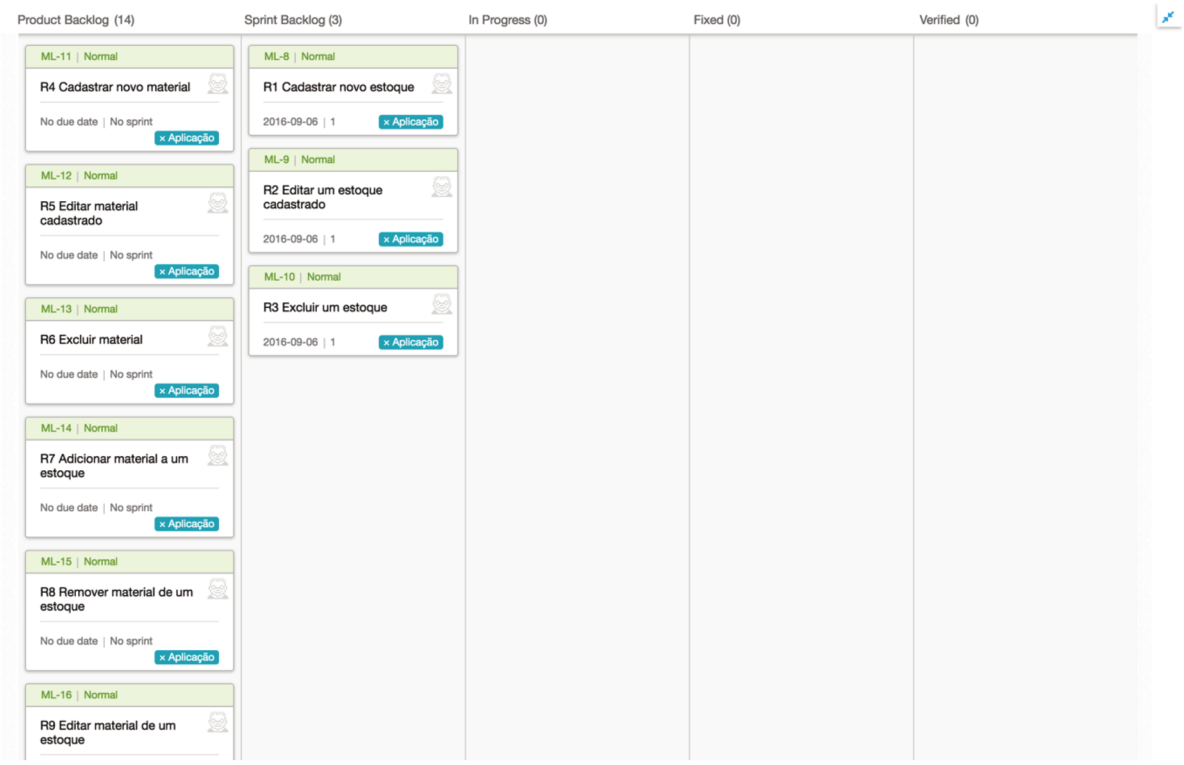
Para o presente trabalho, utilizou-se como metodologia de desenvolvimento de *software* uma adaptação do *framework* Scrum, visto na seção 2.1.1. A adaptação foi necessária uma vez que existe somente um membro no projeto, e não uma equipe, como previsto no Scrum.

A opção foi escolhida visando buscar as vantagens que o *framework* propõe para o gerenciamento de produto: velocidade no desenvolvimento, cliente sempre a par do desenvolvimento do produto, diminuição de bugs, flexibilidade para alteração de prioridades e priorização de atividades que agregam valor.

Com a existência de um único membro, os três papéis são atribuídos para a mesma pessoa, ou seja, a mesma pessoa cumpre as funções de *Product Owner*, de Scrum Master e de time de desenvolvimento. O evento de *Daily Scrum* passa a ser desnecessário neste contexto, porém todos os outros eventos são mantidos, com a única alteração de que não são mais reuniões, mas sim avaliações individuais sobre cada aspecto. Os artefatos também se mantêm inalterados.

No contexto deste trabalho, foi utilizado o software YouTrack para gerenciamento do *Product Backlog* e *Sprint Backlog*. Na figura 4.1, pode-se ver uma captura de tela do estado inicial do primeira *Sprint* no quadro de gerenciamento de tarefas do YouTrack. A primeira coluna contém as tarefas do *Product Backlog*, a segunda as tarefas do *Sprint Backlog*, a terceira as tarefas em progresso, a quarta as tarefas concluídas e a quinta as tarefas verificadas.

Figura 4.1 – Quadro de gerenciamento de tarefas.



Fonte: autora.

O fluxo do Scrum também foi mantido para o cenário individual. Inicialmente, por meio de entrevistas com os colaboradores do LPTox e conversas com o orientador do projeto, foi estabelecido o *Product Backlog*. O *Sprint* foi definido como o período de uma semana, sendo planejada no seu início e sendo revisada e analisada retrospectivamente no seu fim. Ao final de cada *Sprint*, o protótipo até então elaborado era apresentado aos membros do LPTox para avaliação e *feedback*, bem como ao orientador do projeto.

A adaptação do Scrum para um único membro apresentou-se tão vantajosa quanto o convencional, mantendo as ressalvas do contexto em questão.

4.1.2 Estratégia para Levantamento de Requisitos

O levantamento de requisitos, ou levantamento de dados, é o processo de reunir informações sobre os sistemas existentes e sobre o sistema proposto (FIRPO, 2011). Denis Alcides Rezende fala sobre a importância deste processo em seu livro Engenharia de Software e Sistemas de Informação (REZENDE, 2005) “O sucesso no desenvolvimento de um projeto, sistema ou *software*, depende fundamentalmente

do levantamento de dados, em face dessa atividade ser a base que permitirá ao pesquisador ou desenvolvedor tirar conclusões sobre situações, problemas ou fenômenos e, assim, sugerir propostas que possam contribuir para a solução de problemas, criação de sistemas, ou melhoria dos assuntos investigados". Visto tamanha importância, tal processo não poderia deixar de estar presente. Assim, para o levantamento de requisitos desta aplicação foram utilizadas três técnicas diferentes: entrevista, análise de documentos e prototipação, visando garantir o melhor aproveitamento possível.

A entrevista foi a primeira técnica a ser aplicada. O passo inicial consistiu na elaboração de um roteiro para guiar as perguntas aos entrevistados. Tal roteiro foi baseado no artigo de Fernando Firpo "Processos de Obtenção de Requisitos - Parte (1/8)" (FIRPO, 2011). Em seguida, via *e-mail*, agendou-se horários para as entrevistas. Cada entrevista teve a duração de uma hora. As entrevistadas foram a supervisora de laboratório do LPTox, Alcíbia Helena de Azevedo Maia, e a técnica de laboratório do LPTox, Maitê Perin. As entrevistas possibilitaram a criação dos requisitos funcionais do sistema e uma maior compreensão da aplicação. O roteiro, a entrevista com a supervisora e com a técnica de laboratório podem ser encontrados nos anexos A, B e C deste trabalho, respectivamente.

No *e-mail* de agendamento das entrevistas, também foi solicitado que as participantes enviassem todo o tipo de documento que pudesse auxiliar na compreensão do quê é o laboratório e de como funciona o sistema de estoque atual. A solicitação foi feita para possibilitar a análise de documentos, a segunda técnica de levantamento de requisitos aplicada. Foram enviados um total de nove documentos que auxiliaram, principalmente, sanando dúvidas referentes aos requisitos. Com os documentos, muitas questões eram esclarecidas sem a necessidade do contato imediato com os colaboradores do LPTox, mas sim apenas consultando os documentos.

A última técnica aplicada foi a prototipação, um processo iterativo durante todo o desenvolvimento do *software*. Ao final de cada *Sprint*, o protótipo até então criado era apresentado ao orientador do projeto e aos colaboradores do LPTox, em sessões individuais. As sessões consistiam em deixar o usuário livre para o uso da aplicação, sem orientar como esta deveria ser utilizada. Desta maneira, foi possível verificar a usabilidade do *software*, bem como se os requisitos estavam sendo implementados como o desejado. Ao final de todas as sessões, os requisitos eram

atualizados, se necessário, e as modificações necessárias na aplicação eram planejadas para o *Sprint* seguinte. Este método foi fundamental para validar todo o desenvolvimento do projeto durante sua criação, evitando surpresas desagradáveis na entrega final.

Aplicando as três técnicas, foi possível construir um conjunto de requisitos sólido de agrado ao usuário, assim como uma aplicação que reflete tais requisitos da melhor forma possível – uma aplicação bem-sucedida - sempre visando os desejos do usuário final.

4.2 Requisitos funcionais

A aplicação consiste de dezesseis requisitos funcionais que devem ser cumpridos para o sucesso do *software*, segundo os colaboradores do LPTox.

R1 Cadastrar novo estoque

Cadastrar um novo estoque, contendo os seguintes campos para preenchimento: nome, descrição e pessoa responsável.

R2 Editar um estoque cadastrado

Editar os campos de um estoque cadastrado.

R3 Excluir um estoque

Excluir um estoque cadastrado.

R4 Cadastrar novo material

Cadastrar um novo material no sistema. Nome e categoria devem ser campos obrigatórios. Além disso, cada categoria terá seus próprios campos, como mostra a tabela 4.1 abaixo.

Tabela 4.1 – Campos para cadastro de material por categoria.

Categoria	Campos
Utensílios	Código da UFSC Descrição Quantidade: por unidade
Vidrarias	Código da UFSC Descrição Volume: ml
Reagentes	Código da UFSC Descrição CAS Grau de pureza Sinônimo Marca Subcategoria: Líquido Volume: em ml Sólido Peso: em g Formato do material neste estoque: barra, pó ou pastilha (se sólido)
Padrões	Descrição Concentração

Fonte: autora.

R5 Editar material cadastrado

Editar campos de um material cadastrado no sistema.

R6 Excluir material

Excluir um material cadastrado no sistema.

R7 Adicionar material a um estoque

Adicionar um material cadastrado a um estoque existente. Além disso, informar as características específicas daquele material no contexto do estoque em questão. Tais especificidades variam de acordo com categoria e subcategoria, conforme a tabela 4.2.

Tabela 4.2 – Campos para adicionar material em um estoque.

Categoria	Subcategoria	Campos
Utensílios	-	Quantidade do material neste estoque: por unidade
Vidrarias	-	Quantidade do material neste estoque: por unidade
Reagentes	Líquido ou Sólido	Validade do material estocado Estado de conservação do material estocado: ruim, médio, bom
	Líquido	Quantidade do material neste estoque: em ml
	Sólido	Quantidade do material neste estoque: em gramas
Padrões	-	Volume do material neste estoque: (ml ou g) Validade
Todas	-	Usuário responsável pelo material no estoque

Fonte: autora.

R8 Remover material de um estoque

Remover material do estoque.

R9 Editar material de um estoque

Editar as informações específicas do material no contexto do estoque em questão.

R10 Filtrar materiais

O usuário poderá pesquisar materiais que estão em um estoque por meio de filtros, a fim de conseguir ter uma melhor visão dos estoques gerenciados. Os filtros são os seguintes:

- Por nome: retorna todos os materiais com o mesmo nome buscado;
- Materiais acabando: retorna todos os materiais que estão acabando;
- Por validade: retorna todos os materiais que estão quase saindo da validade;
- Por estoque: retorna todos os materiais do estoque buscado;
- Por categoria: retorna todos os materiais da categoria buscada;

- Por subcategoria: retorna todos os materiais da subcategoria buscada;

Ao mostrar os resultados das buscas, o sistema também deve possuir uma opção para gerar um arquivo no formato .xls contendo os resultados apresentados.

R11 Gerar modelo de orçamento a partir de materiais selecionados

O usuário poderá selecionar os materiais cadastrados no sistema que deseja solicitar e, em seguida, o sistema deve gerar um modelo de orçamento com os itens marcados.

R12 Alertar sobre materiais que estejam acabando

Alertar aos usuários, no próprio sistema e via *e-mail*, quando um material está quase acabando.

R13 Alertar sobre materiais que estejam saindo da validade

Avisar aos usuários, no próprio sistema e via *e-mail*, quando um material esteja saindo da validade - apenas para materiais que estiverem habilitados para essa função.

R14 Informar se um material já foi utilizado parcialmente ou não

Para cada material do estoque, informar se ele já foi ou não utilizado.

R15 Cadastrar usuário

Cadastrar um usuário informando nome, *e-mail* e papel (campos obrigatórios).

Cada usuário deverá ter um papel, e cada papel possui suas permissões, conforme a tabela 4.3 abaixo.

Tabela 4.3 – Papel x Permissões.

Papel	Permissões
Supervisor	Tem permissão total à aplicação
Professor	Editar o campo “responsável” de um estoque Ter acesso de leitura de todos os itens Aplicar filtros Gerar relatórios
Técnico	Cadastrar novo estoque Editar um estoque cadastrado Excluir um estoque Cadastrar novo material Editar material cadastrado Excluir material Adicionar material a um estoque Remover material de um estoque Editar material de um estoque Aplicar filtros Gerar relatórios Gerar modelo de orçamento Gerar modelo de memorando
Pesquisa / Monitoria	Ter acesso de leitura aos materiais de sua responsabilidade Remover material de um estoque, se este material estiver sob sua responsabilidade Adicionar material a um estoque, se este material estiver sob sua responsabilidade

Fonte: autora.

R16 Gerar modelo de memorando a partir dos materiais selecionados

O usuário poderá selecionar os materiais cadastrados no sistema que deseja solicitar e, em seguida, o sistema deve gerar um modelo de memorando com os itens marcados.

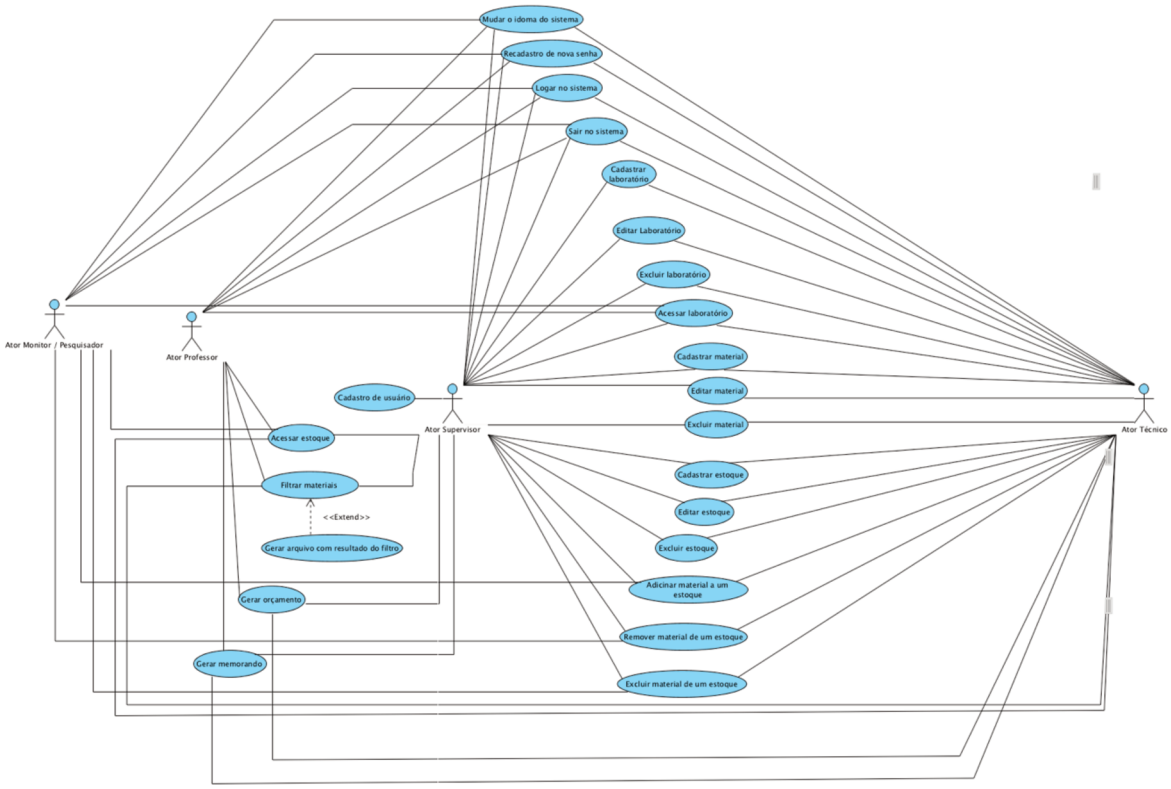
4.2.1 Diagrama de casos de uso

O diagrama de casos de uso documenta quais são as funcionalidades do sistema e como é a interação dos usuários com cada uma dessas funcionalidades, de uma forma simples e bem visual.

Para este contexto, iremos adotar a flecha pontilhada como sendo uma inclusão, ou seja, uma relação em que para o caso de uso ter sua funcionalidade executada precisa chamar outro caso de uso.

Sendo assim, a figura 4.2 nos mostra o diagrama de casos de uso do sistema.

Figura 4.2 – Diagrama de casos de uso.



Fonte: autora.

4.3 Arquitetura da solução

A aplicação em questão foi construída utilizando o *framework* Ruby on Rails 4 – ver seção 5.1.2 – sendo assim, sua arquitetura segue fielmente a arquitetura deste *framework*. Isso porque, uma das características do Ruby On Rails é justamente impor algumas restrições bastante severas sobre como você deve estruturar suas aplicações, mas surpreendentemente, essas restrições tornam muito mais fácil a criação delas (RUBY et al., 2013).

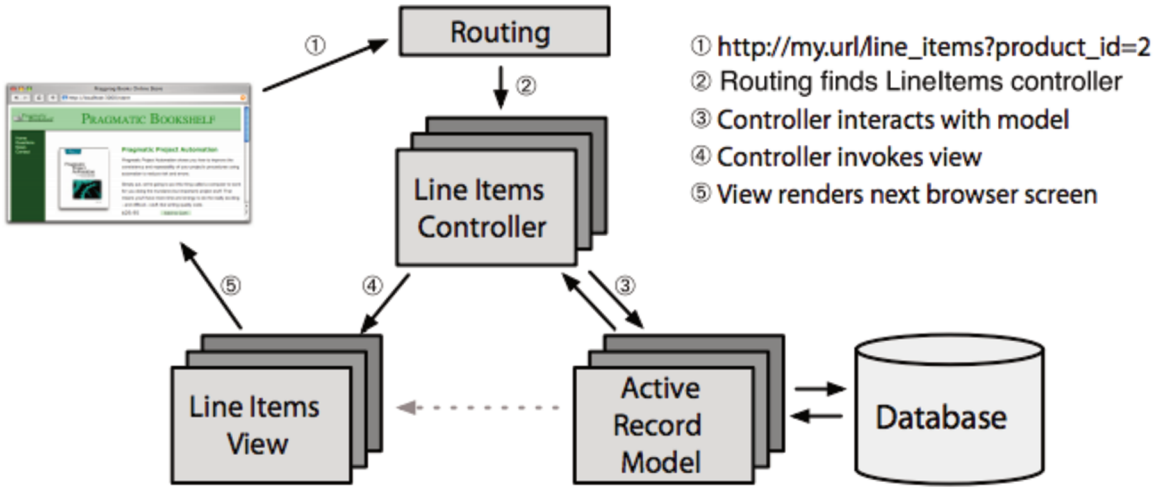
Na seção 2.2 abordou-se a utilização do MVC para implementação da Camada de Aplicação. Além das vantagens já conhecidas do MVC, a decisão por adotar a sua utilização também foi pelo fato do Rails ser um *framework* MVC, isto é, uma de suas restrições é a utilização deste padrão de projeto.

Aplicando o MVC para o Rails, quando um usuário gera alguma requisição através do navegador, essa requisição é primeiramente enviada ao roteador, que é responsável por analisá-la e repassá-la para o controlador apropriado. O roteador deve também indicar qual método deste controlador será incumbido de receber a

requisição - esse tipo de método é conhecido como uma ação do controlador. A ação tem acesso aos dados da requisição, pode interagir com o modelo, pode chamar outras ações e pode, eventualmente, preparar informações para a visão, que exibe algo para o usuário no navegador.

A figura 4.3 refere-se a um exemplo dado pelo livro “Agile Web Development with Rails 4” (RUBY et al., 2013) que ilustra precisamente como a arquitetura Rails funciona e é composta. No exemplo, o aplicativo exibiu anteriormente uma página de catálogo de produtos, e o usuário recém clicou no botão “Adicionar ao carrinho” ao lado de um dos produtos. Este botão realiza um método http POST para http://localhost:3000/line_items?product_id=2, onde line_items é um recurso na aplicação e 2 é o nosso ID interno para o produto selecionado.

Figura 4.3 – Rails e MVC.



Fonte: RUBY, Sam; THOMAS, Dave; HANSSON, David Heinemeier. **Agile Web Development with Rails 4.** 4. ed. Dalas: Rails, 2013.

Neste caso, o roteador recebe a requisição e a repassa para a ação create, do controlador Line Items. Nesta ação, é encontrado o carrinho de compras do usuário atual (que é um objeto gerenciado pelo modelo), são solicitadas ao modelo informações sobre o produto 2 e pede-se ao carrinho para adicionar tal produto a si próprio. Então, o controlador prepara para que a visão tenha acesso ao objeto carrinho e chama o código da visão para mostrar ao usuário o novo produto adicionado.

Caso fosse necessária a criação de um novo requisito no sistema, na maioria dos casos as modificações seriam parecidas, uma vez que é justamente esse intuito do *framework*: automatizar ações que o usuário possivelmente vai fazer. O roteador precisaria ser atualizado, já que ele é único na aplicação, para saber qual a nova rota do novo requisito. Além disso, um controlador novo, uma visão nova e um modelo novo precisariam ser acrescentados. Por fim, o banco de dados também precisaria ser atualizado, caso fosse necessária a persistência dos dados. Todas essas ações podem ser feitas a partir de um único comando do Rails, chamado Scaffold, permitindo maior velocidade e consistência no desenvolvimento.

Dessa forma, seguir a arquitetura do Rails proporciona agilidade, facilidade de desenvolvimento, sendo o Scaffold um exemplo disso, e alta manutenibilidade, uma vez que implementa o MVC.

5 Desenvolvimento

5.1 Tecnologias utilizadas

5.1.1 Ruby

Ruby é uma linguagem de programação interpretada criada pelo Japonês Yukihiro "Matz" Matsumoto e lançada publicamente no ano de 1995 (RUBY, 2016). Escrita em C, segundo David Flanagan e Yukihiro Matsumoto no livro *The Ruby Programming Language* (FLANAGAN; MATSUMOTO, 2008) “Ruby inspira-se em Lisp, Smalltalk e Perl, mas usa uma gramática que é de fácil aprendizado para programadores C e Java™. Ruby é uma linguagem puramente orientada a objetos, mas também se adequa aos estilos de programação procedurais e funcionais”. Matz queria uma linguagem de script que fosse mais poderosa do que Perl, e mais orientada a objetos do que Python (STEWART, 2016).

Pela proposta de alta velocidade de desenvolvimento, por ser orientado a objetos, por ser flexível, aceitando também programação funcional, por ser *open source*, e por possuir um framework, Ruby On Rails (tema de próxima seção), que proporciona um conjunto de ferramentas muito vasto para aplicações web, Ruby foi escolhida como a linguagem de programação para o desenvolvimento desta aplicação.

Ruby está classificada entre as top 10 na maioria dos índices que medem o crescimento e popularidade de linguagens de programação em todo o mundo (RUBY, 2016), comprovando na prática o seu potencial.

5.1.2 Ruby on Rails

Ruby on Rails, ou somente Rails, é um *framework* de desenvolvimento de aplicações web escritas na linguagem Ruby. Ele é projetado para tornar a programação de aplicações web mais fácil, fazendo suposições sobre o quê cada desenvolvedor precisa para começar (BIGG et al., 2016). RUBY e colaboradores (2013) vão além no livro *Agile Web Development with Rails* 4 (RUBY et al., 2013) “Ruby on Rails é um framework que torna mais fácil de desenvolver, implantar e manter aplicações web”.

Por ser um framework *open source*, que promete aumentar velocidade e facilidade no desenvolvimento de sites orientados a banco de dados, exatamente o caso de uma aplicação de gerenciamento de estoque, optou-se por utilizar Ruby on Rails neste projeto.

Basecamp, GitHub, Shopify, Airbnb, Twitch, SoundCloud, Hulu, Zendesk, Square, Highrise são exemplos de aplicações construídas com Rails, mas esses são apenas alguns dos grandes nomes, pois existem literalmente centenas de milhares de aplicativos criados com o *framework* desde o seu lançamento em 2004 por David Heinemeier Hansson (RAILS, 2016). Entende-se, então, com uma decisão sensata a utilização do *framework* no desenvolvimento desta aplicação.

5.1.3 Bootstrap

Para a implementação da visão do projeto, foi utilizado o *framework* Bootstrap. Implementado na empresa Twitter por Mark Otto e Jacob Thornton em meados de 2010, o Bootstrap é voltado inteiramente para a visão das aplicações, o *front-end*. Como a própria página do projeto descreve: “Bootstrap é o mais popular *framework* HTML, CSS, e JS para desenvolvimento de projetos responsivo e focado para dispositivos móveis na web” (OTTO et al., 2016).

O Bootstrap fornece definições básicas de estilo para todos os componentes chave HTML por meio de um vasto conjunto de folhas de estilo. Assim, consegue-se uma aparência uniforme e moderna para aplicações web. Além disso, também possui elementos JavaScript na forma de *plugins* jQuery. Os *plugins* fornecem componentes novos, como carrosséis, e também estendem as funcionalidades de alguns já existentes, como a função de auto completar para campos de entrada.

Optou-se por utilizar Bootstrap nesse projeto por ser *open source*, proporcionar design responsivo, possuir ótimo sistema de grid, por tornar o desenvolvimento da visão da aplicação muito mais ágil, por possibilitar a reutilização de componentes, fornecendo maior manutenibilidade e principalmente, por facilitar a criação de interfaces gráficas uniformes, consistes e modernas.

5.2 Implementação

A implementação da aplicação foi feita, conforme mencionada na seção 4.1.1, aplicando o *framework* Scrum para um único membro. Esta seção mostra como o Scrum foi aplicado para o desenvolvimento deste *software* e o resultado de cada *Sprint* realizado.

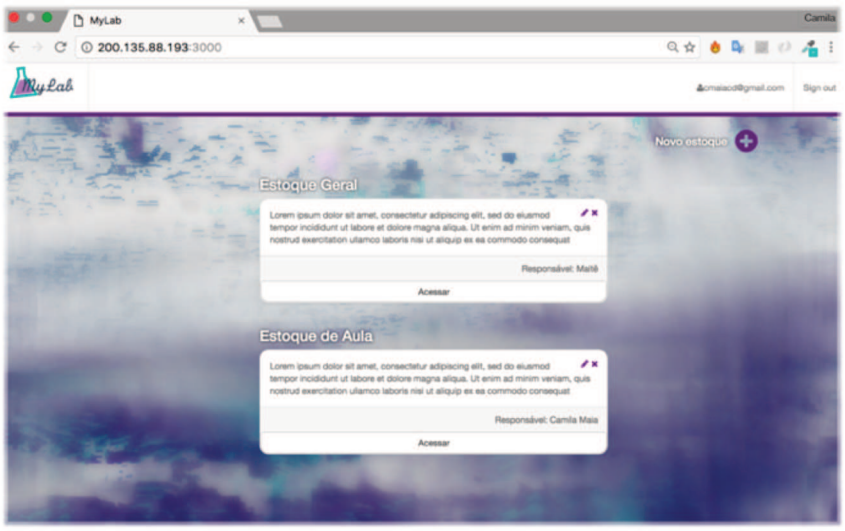
Por a aplicação ser basicamente orientada orientados a banco de dados, optou-se por mostrar em cada *Sprint* o passa a passo da criação das entidades e relacionamentos do sistema. Porém, vale ressaltar que o diagrama de classe se encontra no anexo F deste documento, o qual é fundamental para manutenibilidade do sistema.

5.2.1 *Sprint* 1: Estoque e Usuário

O planejamento do primeiro *Sprint* resultou na adição de quatro itens no *Sprint backlog*: cadastrar novo estoque; editar um estoque cadastrado; excluir um estoque; autenticação de usuário – que deveriam ser implementados até o final do *Sprint*.

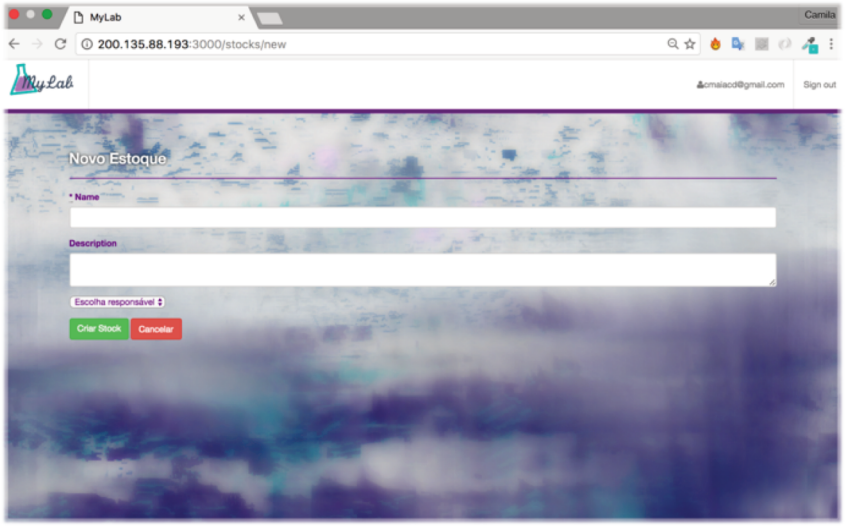
Ao final do processo, foram desenvolvidas cinco telas: tela de *login*, tela de cadastro de usuário, tela inicial, tela de cadastro de estoque e tela de edição de cadastro de estoque. Nas figuras 5.1 e 5.2, pode-se observar como a interface gráfica foi apresentada.

Figura 5.1 – Tela inicial do primeiro protótipo.



Fonte: autora.

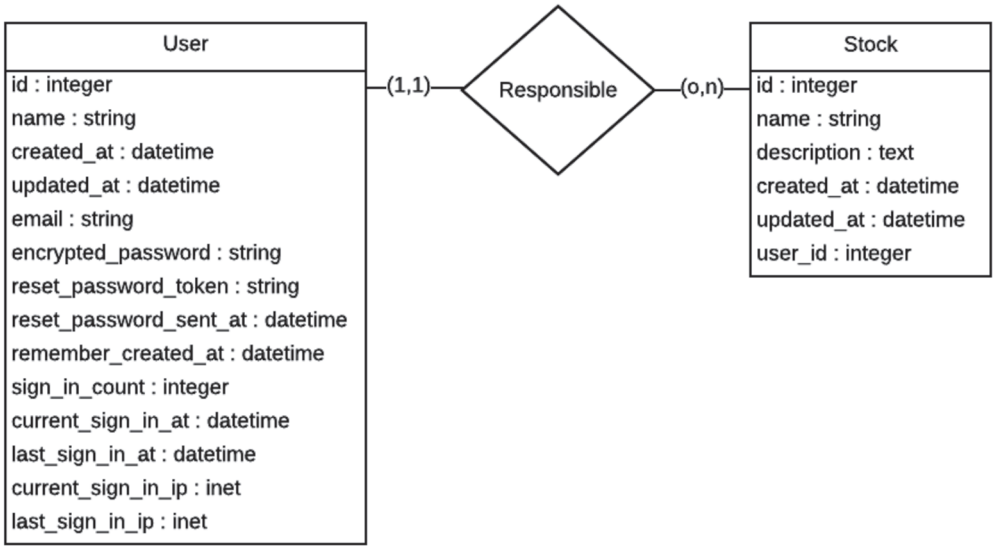
Figura 5.2 – Tela de cadastro de estoque do primeiro protótipo.



Fonte: autora.

Até este ponto, o protótipo apresenta apenas duas entidades: *User*, em português Usuário e *Stock*, em português Estoque. Um usuário pode ser responsável por zero ou muitos estoques, como mostra o modelo Entidade Relacionamento na figura 5.3.

Figura 5.3 – Modelo Entidade Relacionamento do primeiro protótipo.



Fonte: autora

Para o desenvolvimento da autenticação do usuário foi utilizada uma solução gratuita já existente no mercado, a biblioteca Devise (PLATAFORMATEC, 2016). Devise é uma solução de autenticação flexível para Rails criada pela Plataformatec

(PLATAFORMATEC, 2016). O Devise foi o responsável por adicionar muitos dos atributos presentes na entidade Usuário, a fim de promover a autenticação e o controle de sessão no sistema. Com o Devise, foi possível também criar as telas de autenticação e cadastro de usuário sem precisar programar nada referente à interface gráfica.

O protótipo foi testado por três usuários diferentes, sendo que dois deles trabalham no LPTox. Como resultado, foram levantados dois quesitos para melhoria: o botão de excluir estoque traz uma ação de risco, portanto deve estar em um local mais escondido, afim de evitar perdas de dados não desejadas – se possível acrescentar mais algum outro tipo de confirmação da exclusão; a ordem dos estoques muda conforme é feita alguma edição em algum dos estoques – a ordem deveria ficar fixa, de acordo com a data de criação.

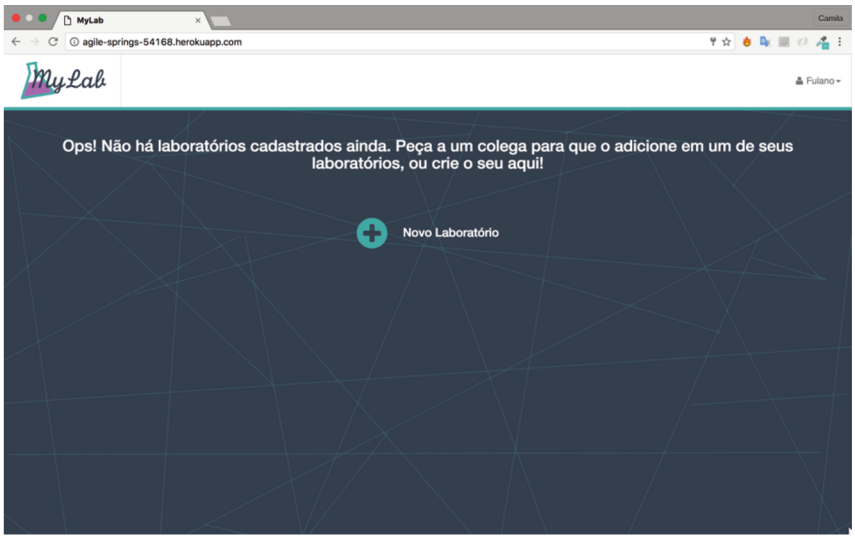
Assim, como retrospectiva do *Sprint* 1, analisou-se que os objetivos planejados foram alcançados e que o protótipo, de forma geral, agradou aos usuários.

5.2.2 *Sprint* 2: Laboratório e Ambiente de Produção

O planejamento do segundo *Sprint* resultou na adição de também três novos itens no *Sprint Backlog*: cadastrar um laboratório; acessar a aplicação em um ambiente de produção; possibilitar a troca de senha de usuário via *e-mail* através do *link* “esqueci minha senha” – além da necessidade de resolver os problemas encontrados pelos usuários no *Sprint* anterior.

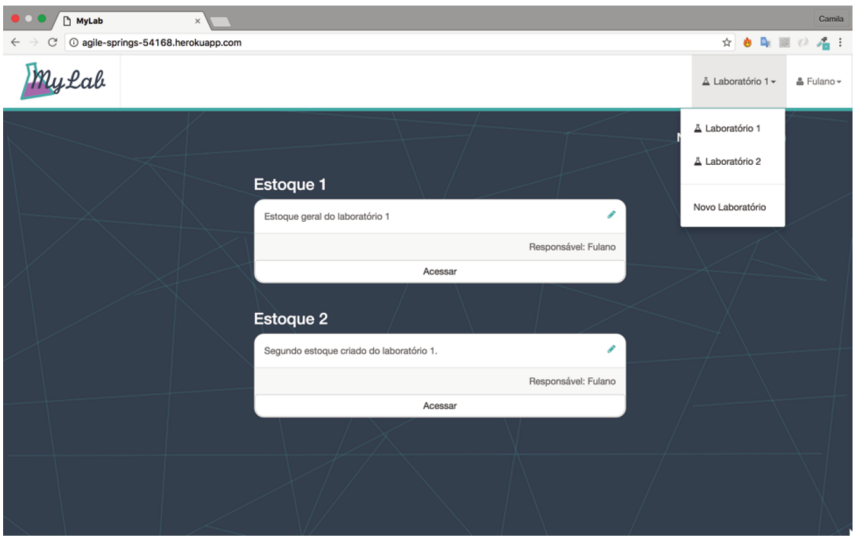
Ao final do processo, foram desenvolvidas três novas telas: tela de cadastro de laboratório, tela de edição de cadastro de laboratório e a tela inicial para quando não há laboratórios cadastrados. Além disso, a barra de menu foi modificada para conter um *dropdown* com as opções de laboratórios para aquele usuário. Nas figuras 5.4 e 5.5, pode-se observar como a interface gráfica foi apresentada.

Figura 5.4 – Tela inicial, quando não há laboratórios cadastrados, do segundo protótipo.



Fonte: autora.

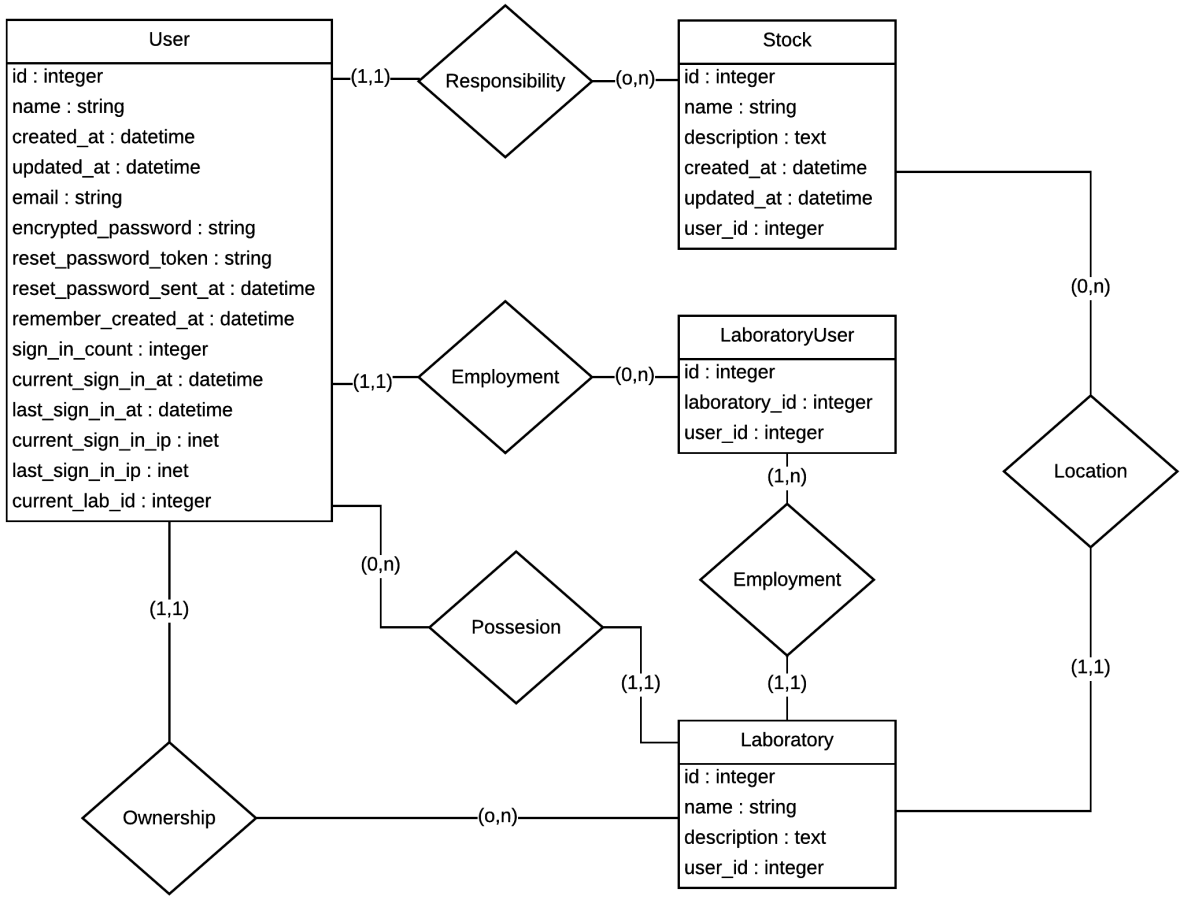
Figura 5.5 – Tela inicial, com o *dropdown* de laboratórios aberto, do segundo protótipo.



Fonte: autora.

Para o desenvolvimento do requisito “cadastrar laboratório”, foi necessária a adição de duas novas entidades no modelo Entidade Relacionamento: *Laboratory*, em português Laboratório, e *LaboratoryUser*, da união de laboratório e usuário. Um usuário pode ser dono de zero ou mais laboratórios e um laboratório sempre possui um único dono; um usuário possui um laboratório atual e um laboratório pode ser o atual de zero ou muitos usuários; um usuário trabalha para um ou mais laboratórios e um laboratório tem um ou mais usuários como colaboradores; um estoque está localizado em um laboratório e um laboratório abriga zero ou mais estoques. As entidades e relações citadas podem ser vistas na imagem 5.6.

Figura 5.6 – Modelo Entidade Relacionamento do segundo protótipo.



Fonte: autora.

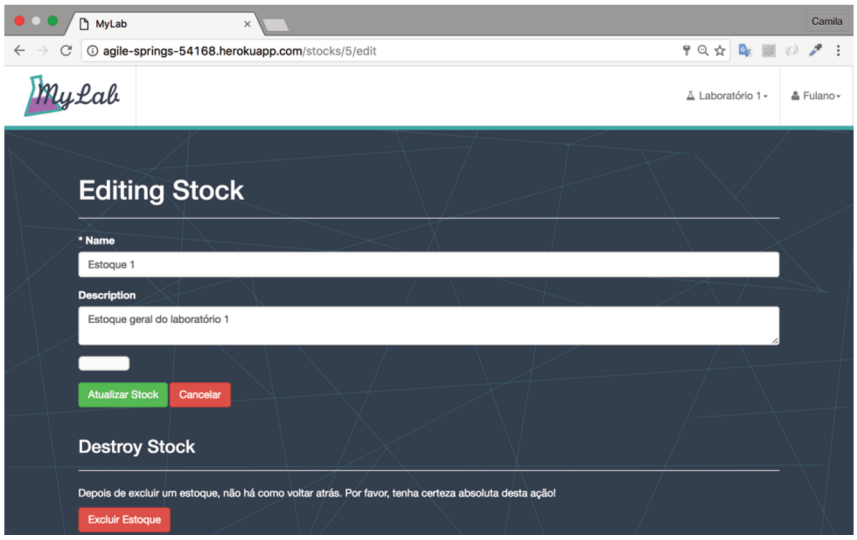
O requisito de possibilitar o acesso à aplicação em modo de produção tem como objetivo apresentar o protótipo aos usuários em um modo mais estável do que o de desenvolvimento, com um link fixo e permitindo o acesso de qualquer lugar. Para isso, foi utilizada uma solução de Plataforma como Serviço, no inglês *Plataform as a Service* (PaaS), chamada Heroku (HENRY et al., 2016). O Heroku permite a implantação de aplicações web, possuindo suporte para várias linguagens, incluindo Ruby, A configuração foi feita inteiramente seguindo as instruções presentes no artigo “Getting Started with Rails 4.x on Heroku” (HEROKU DEV CENTER, 2016), presente no próprio site do Heroku.

O terceiro requisito, responsável por possibilitar a troca de senha de usuário via *e-mail* através do *link* “esqueci minha senha”, foi implementado também utilizando as instruções de um artigo, no caso “*Setting Up Mailer Using Devise For Forgot Password*” (RUBY ON RAILS HELP, 2016). O artigo explica passo a passo

como configurar sua aplicação, tanto em ambiente de desenvolvimento quanto em produção, a fim de possibilitar o envio de mensagens de *e-mail* aos usuários. Ainda, uma conta de *e-mail* do gmail foi criada para ser utilizada no envio das mensagens: mylabassistance@gmail.com, evitando a utilização de uma conta pessoal.

Com relação aos pedidos de mudanças dos usuários do último *Sprint*, os estoques passaram a ser ordenados por id e a ação de remoção de estoque passou a estar presente apenas na tela de edição, com um aviso sobre a importância desta ação, conforme mostra a figura 5.7.

Figura 5.7 – Tela de edição de estoque do segundo protótipo.



Fonte: autora.

Ao final do *Sprint*, o protótipo foi testado novamente pelos mesmos três usuários do *Sprint* anterior. Como resultado, foi levantado um quesito para melhoria: a caixa de seleção de usuário responsável, nas telas de cadastro e edição de estoque, apareceu branca com letras brancas, impossibilitando a leitura das opções para duas das pessoas que testaram, em dois navegadores diferentes.

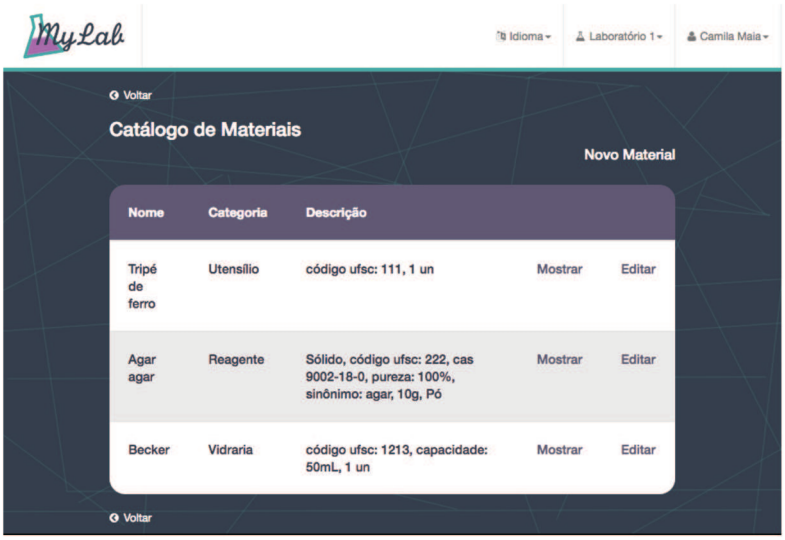
Como retrospectiva do *Sprint* 2, analisou-se que os objetivos planejados foram alcançados e que o protótipo, de forma geral, agradou aos usuários. Os usuários mostraram estar satisfeitos com a usabilidade da aplicação e com as funcionalidades apresentadas até então.

5.2.3 *Sprint* 3: Material e Internacionalização

O planejamento do terceiro *Sprint* resultou na adição de quatro novos itens no *Sprint Backlog*: cadastrar novo material; editar material cadastrado; excluir material; internacionalização do sistema – além da necessidade de resolver os problemas encontrados pelos usuários no *Sprint* anterior.

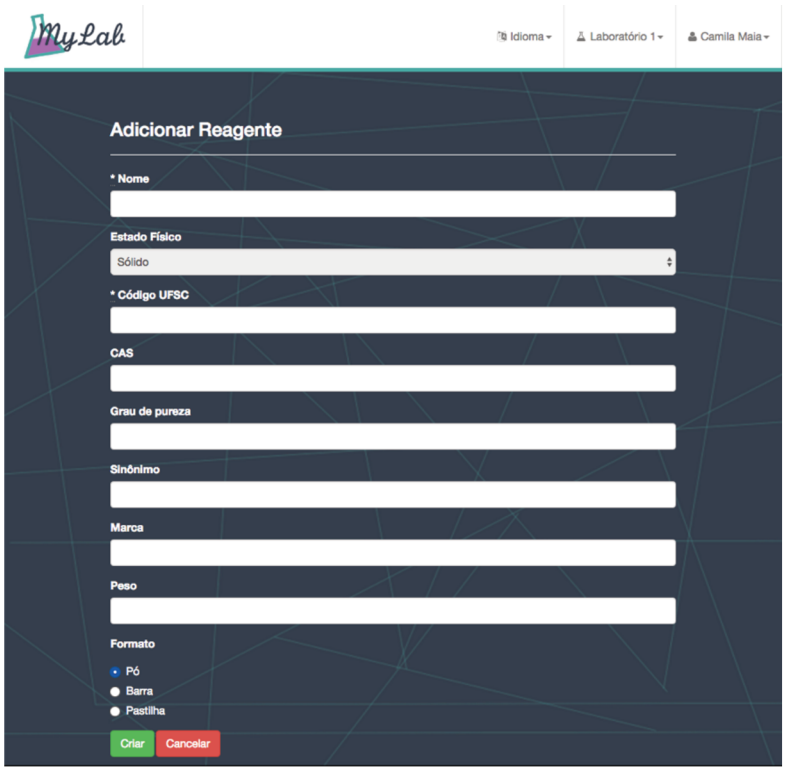
Ao final do processo, foram desenvolvidas dezesseis novas telas: tela de catálogo de materiais e, para cada categoria e subcategoria, uma tela de cadastro de novo material, uma de edição de material cadastrado e uma de exibição do material cadastrado. Além disso, foi adicionado um novo item na barra de menu contendo as opções de idioma do sistema. Nas figuras 5.8 e 5.9, pode-se observar como a interface gráfica foi apresentada.

Figura 5.8 – Tela de catálogo de materiais.



Fonte: autora.

Figura 5.9 – Tela de novo reagente sólido.

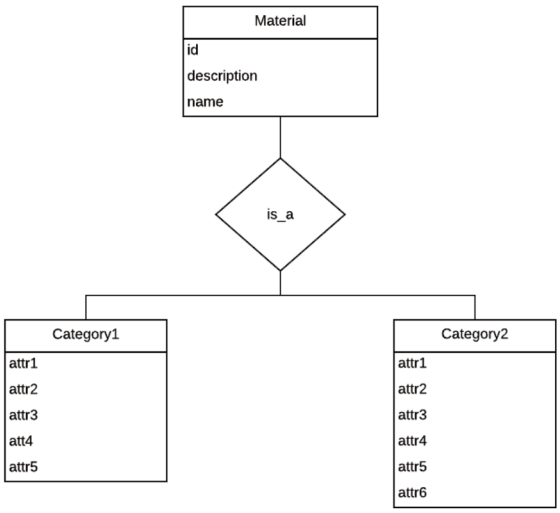


Fonte: autora.

Levando em consideração os requisitos da aplicação, é fácil avaliar que existem materiais de diferentes categorias, os quais possuem diferentes atributos de acordo com sua categoria. Sendo assim, para a implementação das funções de cadastro, edição e exclusão de material para cada categoria, foram analisadas quatro possíveis modelagens de dados: herança com múltiplas tabelas, matriz esparsa, modelo entidade-atributo-valor (do inglês, *entity–attribute–value model* - EAV) e modelagem não relacional.

A herança com múltiplas tabelas propõe a existência de uma entidade chamada Material, contendo os atributos em comum de todas as categorias, ligada pela associação “é um” com uma entidade para cada categoria contendo os atributos específicos, conforme a imagem 5.10.

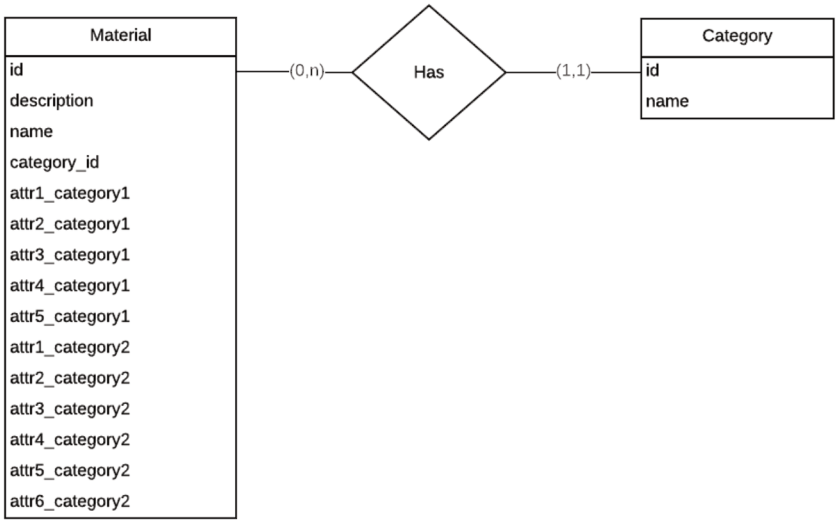
Figura 5.10 – Modelagem através de herança com múltiplas tabelas.



Fonte: autora.

A modelagem a partir da ideia de uma matriz esparsa propõe que exista uma entidade Material que contém todos os atributos de todas as categorias. Essa entidade associada a uma entidade categoria. Portanto, quando um material for da categoria 1, por exemplo, ele possuirá apenas valores para os atributos que esta categoria possui, sendo analogamente comparado a uma matriz esparsa.

Figura 5.11 – Modelagem baseada em matriz esparsa.



Fonte: autora.

O modelo EAV surge para resolver o problema das matrizes esparsas, em que o número de atributos de uma entidade é potencialmente alto, porém o número que é efetivamente aplicado para cada objeto é relativamente modesto. É um paradigma alternativo ao modelo de entidade relacionamento, em que entidade-atributo-valor representa uma tabela que possui três colunas: uma para o tipo de entidade que deve ser representado, outra para o atributo dessa entidade e um terceiro para o valor real do atributo (SEN, 2016).

Utilizando o exemplo do artigo "*Five Simple Database Design Errors You Should Avoid*" (SEN, 2016), tendo a tabela Empregados, figura 5.12, com as colunas número do empregado, primeiro nome, sobrenome e data de nascimento, ao utilizar o modelo EAV, organiza-se os dados a fim de representar os atributos como valores em uma coluna e os valores correspondentes desses atributos em outra coluna. No caso, na figura 5.13, na modelagem EAV passaria a existir uma tabela chamada EmpregradosValores, com as colunas número do empregado, propriedade do empregado e valor.

Figura 5.12 – Exemplo modelagem EAV parte 1.

Employees			
emp_nbr	first_name	last_name	date_of_birth
101	David	Bristol	11/2/1946
102	Mary	Manning	10/10/1951
103	Alistair	Ross	5/21/1966
104	Tom	Snyder	8/4/1959

Fonte: https://www.simple-talk.com/sql/database-administration/five-simple-database-design-errors-you-should-avoid/

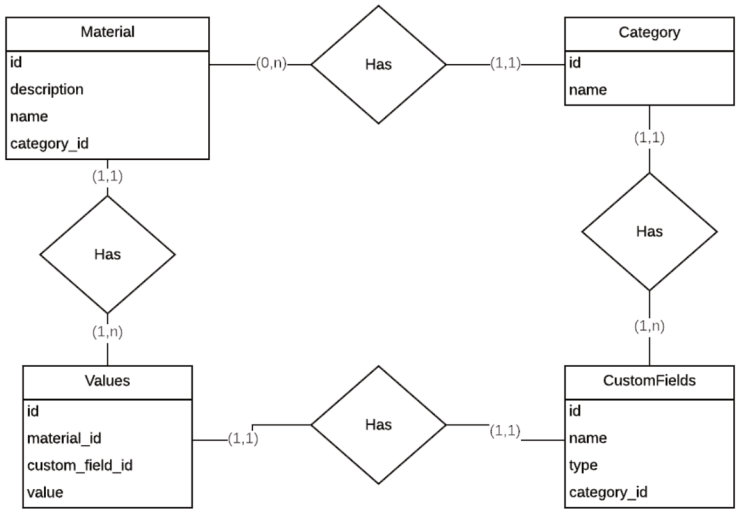
Figura 5.13 – Exemplo modelagem EAV parte 2.

EmployeeValues		
emp_nbr	emp_property	value
101	first_name	David
101	last_name	Bristol
101	date_of_birth	11/2/1946
102	first_name	Mary
102	last_name	Manning
102	date_of_birth	10/10/1951
103	first_name	Alistair
103	last_name	Ross
103	date_of_birth	5/21/1966

Fonte: https://www.simple-talk.com/sql/database-administration/five-simple-database-design-errors-you-should-avoid/

Adaptando o modelo EAV para este contexto, a modelagem do problema consistiria em 4 tabelas: Material, Categoria, CamposCustomizáveis e Valores, conforme mostra a figura 5.14.

Figura 5.14 - Modelagem EAV.



Fonte: autora.

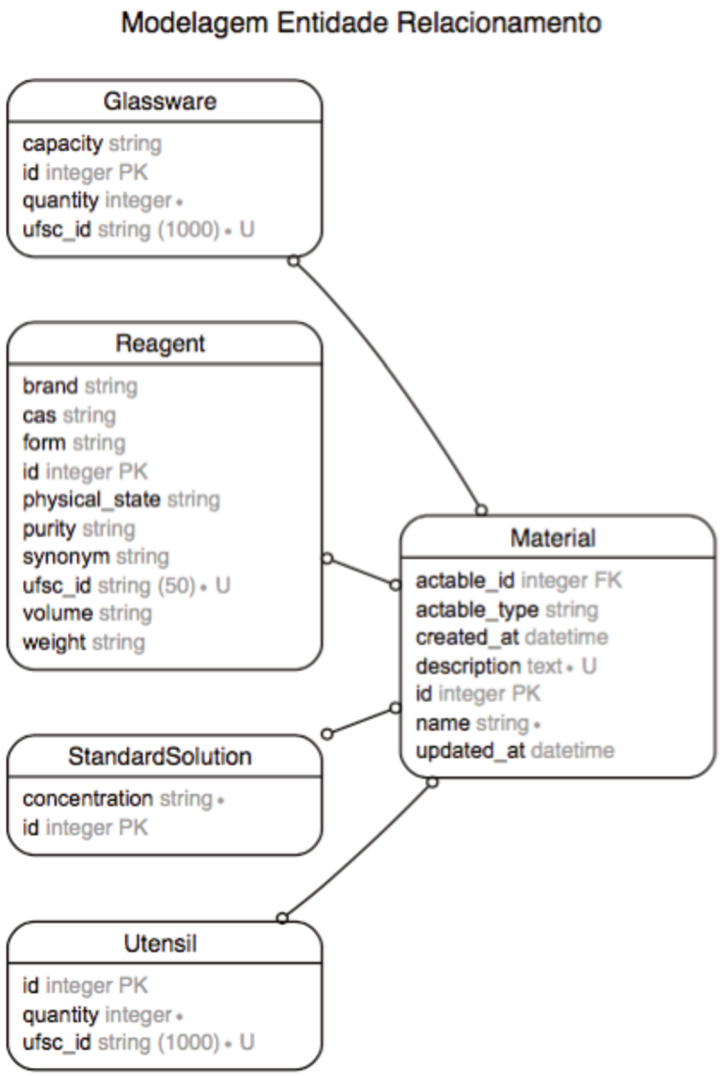
A quarta e última opção levantada foi a utilização de uma modelagem não relacional, caracterizada pela não utilização de esquemas de tabela fixa.

A opção de matriz esparsa tem como vantagem a possibilidade de criação de novas categorias dinamicamente pelo usuário, com atributos e tipos previamente

definidos no banco, porém gera uma tabela muito maior que o necessário, com vários espaços de valor nulo. Além da necessidade de verificação a todo momento na aplicação se o material possui ou não um dado atributo. Já a modelagem EAV permite a criação de novas categorias dinamicamente pelo usuário sem ocupar as tabelas com valores nulos. Entretanto, possui como desvantagem o difícil acesso aos atributos no decorrer da aplicação. Tão grave quanto, os valores reais dos atributos ficam gravados no banco sempre como *string*, sendo necessária a conversão para o real tipo em vários momentos na aplicação. A modelagem não relacional implicaria na mudança de todo o banco de dados, uma vez que até então o banco utilizado era relacional. Apesar desta modelagem ser muito mais flexível e poder proporcionar a adição dinâmica de categorias, não foi avaliado que seria justificável a mudança de paradigma, ponderando o custo da mudança, o porte da aplicação e os benefícios que ela traria. Portanto, a opção adotada foi a herança com múltiplas tabelas, que apresenta como vantagens o uso apropriado da herança e o fácil acesso aos atributos dentro da aplicação. No entanto, a criação dinâmica de novas categorias é impossibilitada. A maneira encontrada para contornar o problema foi a criação de um documento com instruções para adição de uma nova categoria via linha de comando e edição de arquivos. Este documento pode ser encontrado no anexo E deste trabalho.

Para a implementação da solução através de herança com múltiplas tabelas foi preciso utilizar a biblioteca *Active Record Acts As* (ZAMANIN, 2016), uma vez que esse paradigma não é nativo em Rails. O quê precisou ser feito para implantar a biblioteca foi adicionar a palavra “actable” no modelo Material e "acts_as :material" nos modelos de cada categoria. Adicionalmente, foi necessário incluir as colunas “actable_id” e “actable_type” na tabela Material. Assim, modelo entidade relacionamento final deste cenário pode ser visto na imagem 5.15.

Figura 5.15 - Modelo Entidade Relacionamento do terceiro protótipo.



Fonte: autora.

Nota-se ainda, que na figura 5.15, os atributos que representam medidas são do tipo *string*, isto porque está sendo utilizada uma biblioteca Ruby para manipulação de unidades chamada Ruby Units (OLBRICH, 2016), a qual utiliza por padrão unidades como *string*.

Para o próximo requisito do *Sprint*, a internacionalização do sistema, foi utilizada a própria solução já existente no *framework* Rails: Rails Internationalization (I18n) API (FUCHS, 2016). Em poucos passos foi possível disponibilizar ao usuário duas opções de idioma: inglês e português, as quais podem ser escolhidas por meio de um *dropdown* na barra de menu.

Com relação aos pedidos de mudanças dos usuários do último *Sprint*, a caixa de seleção de usuário responsável, nas telas de cadastro e edição de estoque, passou a ter letras visíveis para todos os navegadores, apenas com ajustes de CSS.

Ao final do *Sprint*, o protótipo foi testado novamente pelos mesmos três usuários do *Sprint* anterior. Como resultado, foi levantado um quesito para melhoria: remover o redirecionamento para a tela de exibição do material sempre que houver a criação ou edição de um material. O sugerido foi que o redirecionamento, nesses casos, deve ser feito para a tela de catálogo de materiais, visando maior agilidade na inserção e edição de materiais.

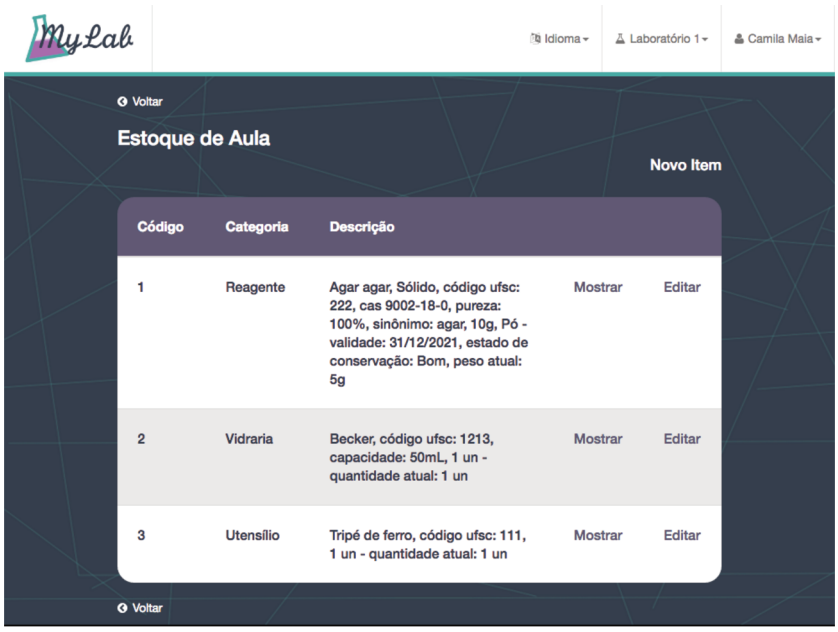
Como retrospectiva do *Sprint* 3, analisou-se que os objetivos planejados foram alcançados e que o protótipo, de forma geral, agradou aos usuários. Os usuários continuaram a mostrar estarem satisfeitos com a usabilidade da aplicação e com as funcionalidades apresentadas até então.

5.3.3 *Sprint* 4: Material em Estoque e Domínio da UFSC

O planejamento do quarto *Sprint* resultou na adição de quatro novos itens no *Sprint Backlog*: adicionar um material a um estoque; remover um material de um estoque; editar material de um estoque, acessar a aplicação através de um domínio da UFSC – além da necessidade de resolver os problemas encontrados pelos usuários no *Sprint* anterior.

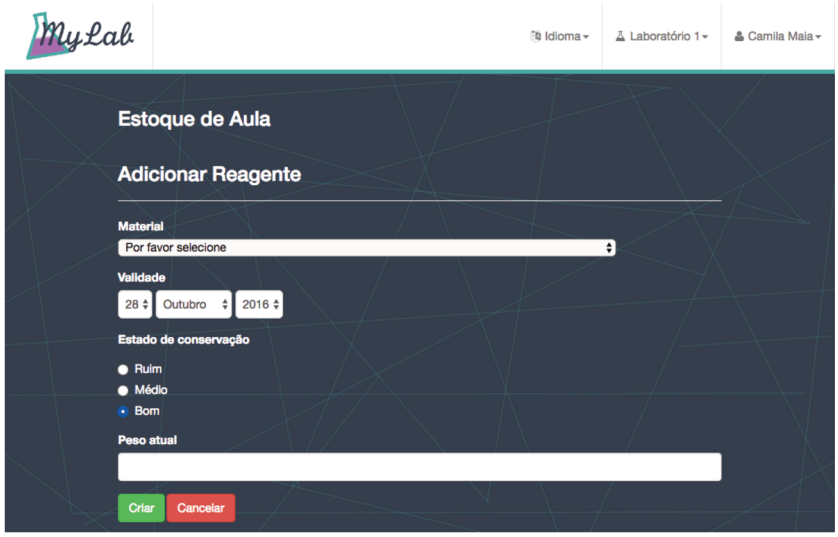
Ao final do processo, foram desenvolvidas dezesseis novas telas: tela de acesso ao estoque e para cada categoria e subcategoria, uma tela de cadastro de novo item do estoque, uma de edição de item do estoque e uma de exibição do item do estoque. Nas figuras 5.16 e 5.17 pode-se observar como a interface gráfica foi apresentada.

Figura 5.16 – Tela de acesso ao estoque.



Fonte: autora.

Figura 5.17 – Tela de novo item de reagente sólido.

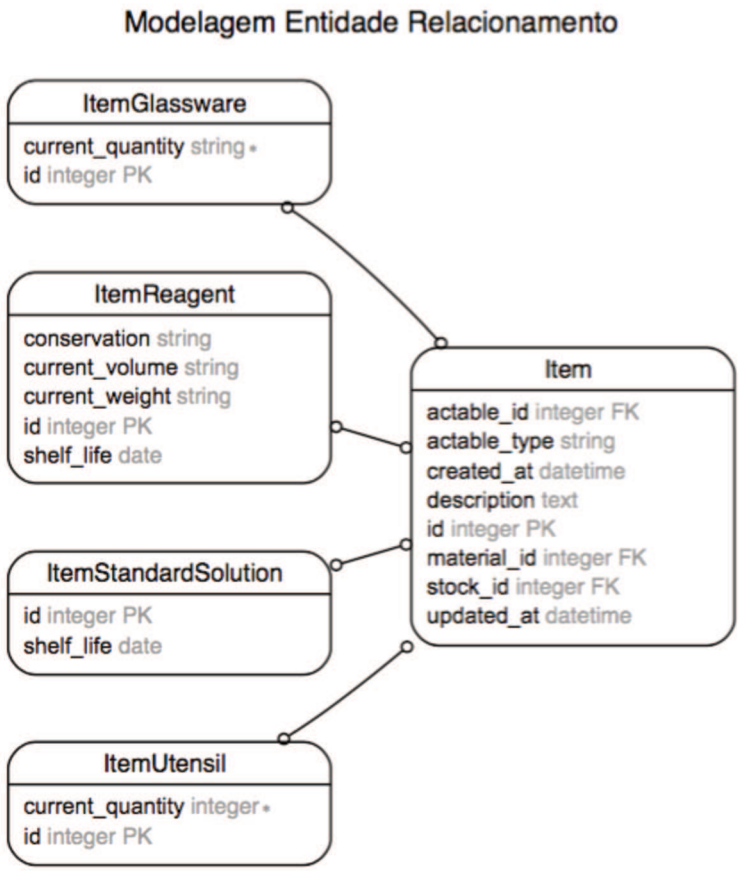


Fonte: autora.

Para a implementação dos requisitos adicionar um material a um estoque, remover um material de um estoque e editar material de um estoque foi adotada a mesma solução para o problema do *Sprint* 3: herança com múltiplas tabelas em conjunto com a biblioteca *Active Record Acts As* (ZAMANIN, 2016). Aqui, foi criada a entidade Item, que representa um material de um estoque, além de uma entidade de item para cada categoria. Na figura 5.18 pode-se ver o diagrama de Entidade

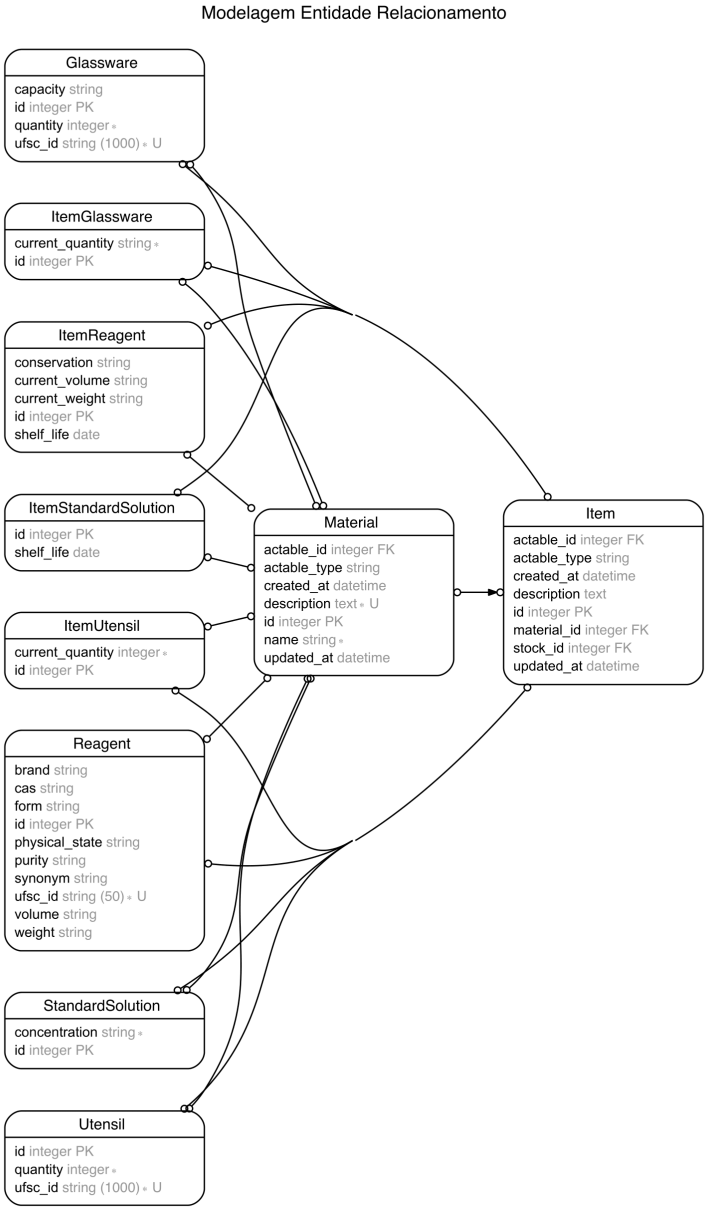
Relacionamento para esse contexto e na figura 5.19 para o contexto que engloba materiais e itens.

Figura 5.18 - Modelo Entidade Relacionamento do quarto protótipo – item.



Fonte: autora.

Figura 5.19 - Modelo Entidade Relacionamento do quarto protótipo – item e material.



Fonte: autora.

O requisito acessar a aplicação através de um domínio da UFSC foi implementado com a utilização da tecnologia Docker (DOCKER, 2016). Um servidor da UFSC foi disponibilizado para o projeto e a aplicação pode ser acessada através do link www.mylab.ufsc.br. As instruções de instalação e implantação do sistema se encontram no anexo D deste documento.

Com relação aos pedidos dos usuários do último *Sprint*, os redirecionamentos foram adaptados como sugerido.

Ao final do *Sprint*, o protótipo foi testado novamente pelos mesmos três usuários do *Sprint* anterior. Como resultado, foram levantados alguns quesitos para melhoria: melhorar a usabilidade para os campos quantidade – não ficou claro que a quantidade deve ser diferente de um somente para casos de caixas ou pacotes: antes da remoção de um material, verificar se existe algum item apontando para este material; acrescentar ícone do sinal de mais e seta para baixo ao lado das ações de acrescentar novo material e novo item; alargar a coluna do nome nas tabelas que listam os itens;

Como retrospectiva do *Sprint* 4, analisou-se que os objetivos planejados foram alcançados e que o protótipo, de forma geral, agradou aos usuários. Os usuários mostraram estar incomodados com pequenos pontos da usabilidade da aplicação, porém no contexto geral obteve-se satisfação com as funcionalidades apresentadas até então.

5.4 Avaliação

Com a implantação da aplicação tanto no servidor do Heroku como no servidor da UFSC, o protótipo pode ser testado pela supervisora e pela técnica de laboratório do LPTox ao longo das etapas de desenvolvimento.

Mesmo sendo ainda um protótipo, em entrevista com a supervisora do LPTox, ela afirma que, com as funcionalidades atuais que o *software* apresenta, tem-se 70% dos casos de uso cobertos. Com a implementação do filtro de materiais, esse número já atingiria a casa dos 90%.

6 Conclusão

Com o presente trabalho pode-se perceber que o gerenciamento de estoque é um problema recorrente e importante para laboratórios de química. Que a ausência de um controle adequado de estoque pode levar a perda de dinheiro, de materiais e de tempo. Conclui-se também que aplicações webs podem sim ajudar a resolver esse tipo de situação. Inclusive, que já existem aplicações presentes no mundo acadêmico e no mercado com esse objetivo. Por outro lado, tais soluções não atendem a todas as necessidades específicas que o LPTox apresentou, sendo então de relevância o desenvolvimento do presente projeto.

O levantamento de requisitos utilizando as três técnicas: entrevistas, análise de documentação e prototipação foi visto como fundamental para o fornecimento de conhecimento e embasamento para a criação de um conjunto de requisitos sólidos, que foram sendo adaptados ao longo de todo o processo de desenvolvimento do trabalho. Analisou-se, também, que a metodologia adotada, o *framework* Scrum, proporcionou como benefícios a iteratividade, organização, velocidade, e garantiu que os futuros usuários fossem aos poucos avaliando os protótipos criados, evitando surpresas desagradáveis ao final de todo o processo. Também, que a utilização do *framework* Rails para o desenvolvimento da aplicação permitiu agilidade na codificação, uma vez que ele é próprio para o desenvolvimento de sites orientados a banco de dados, que é exatamente o caso de uma aplicação de gerenciamento de estoques.

Através do *feedback* contínuo dos colaboradores do LPTox e da avaliação do protótipo atual, que constatou como 70% dos casos de uso já concluídos, conclui-se que a aplicação já pode trazer benefícios reais para o laboratório. Também, que ainda necessita de complementação, principalmente a implementação do requisito de filtragem. Com isso, avalia-se que o projeto já cumpriu com grande parte de seus objetivos traçados, e que se pretende finalizar os que ainda faltam.

6.1 Trabalhos Futuros

Como trabalhos futuros sugere-se primordialmente a continuidade da implementação dos requisitos faltantes. Imediatamente depois, entende-se que a

implementação de testes de integração e de unidade precisariam ser implementados a fim de garantir maior consistência e manutenibilidade da aplicação.

Adicionalmente, como visto na seção 3, a importação de dados utilizando documentos no formato xls poderia ser uma funcionalidade muito útil ao programa, uma vez que evitaria o cadastro de materiais um a um pelo usuário, possibilitando o cadastro em blocos. Existem também opções que focam tornar o sistema um pouco mais inteligente, como previsões de quando um material pode acabar com base no histórico de transações. Ou ainda, análises periódicas das transações realizadas exibidas por meio de gráficos, como por exemplo: quanto foi utilizado do reagente x no último mês ou qual estoque foi mais acessado na última semana.

Por último, entende-se como interessante a criação de uma *Application Program Interface* (API) para consumo dos dados do sistema. Assim, seria possível realizar consultas específicas ao banco de dados e também que outras aplicações consumissem esses dados já coletados.

Referências

BIGG, Ryan et al. **Getting Started with Rails.** Disponível em: <http://guides.rubyonrails.org/getting_started.html#what-is-rails-questionmark>. Acesso em: 14 out. 2016.

CARVALHO, Nelio Garbellini de; CHAGAS, Thiago Augusto de Castro; MACHADO, Ana Marta Ribeiro. Implantação de um sistema de gestão de reagentes em laboratórios universitários. **Augmdomus,** Montevideo, v. 2, p.72-81, dez. 2010.Disponível em:< http://www.revistas.unlp.edu.ar/domus/article/viewFile/134/164 >. Acesso em: 18 out 2016.

CHAK, Dan. **Enterprise Rails.** Sebastopol, Califórnia, EUA: O'reilly, 2009.

CONALLEN, Jim. **Building Web Applications with UML**. 2. Ed. Disponível em:<https://www.pearsonhighered.com/samplechapter/0201730383.pdf>. Acesso em 07 out. 2016.

DOCKER (Califórnia, EUA). **Docker.** Disponível em: <https://docs.docker.com/>. Acesso em: 30 out. 2016.

DOCKER (São Francisco, Califórnia, EUA). **Quickstart: Docker Compose and Rails.** Disponível em: <https://www.docker.com/company/contact>. Acesso em: 30 out. 2016.

EASLEY, David; KLEINBERG, Jon. **Networks, Crowds, and Markets:** Reasoning about a Highly Connected World. Cambridge: Cambridge University Press, 2010. Disponível em: <https://www.cs.cornell.edu/home/kleinber/networks-book/networks-book.pdf>. Acesso em: 25 ago. 2016.

FIRPO, Fernando. **Análise de Requisitos**: Processos de Obtenção de Requisitos. 2011. Disponível em: <http://analiserequisitos.blogspot.com.br/2011/07/processos-de-obtencao-de-requisitos.html>. Acesso em: 2 jul. 2016

FLANAGAN, David; MATSUMOTO, Yukihiro. **The Ruby Programming Language.** Df: O'reilly, 2008. 444 p. Disponível em: <http://haris-krajina.rhcloud.com/wp-content/uploads/2013/01/baba.pdf>. Acesso em: 05 out. 2016.

FUCHS, Sven; MINAŘÍK, Karel. **Rails Internationalization (I18n) API.** Disponível em: <http://guides.rubyonrails.org/i18n.html#authors>. Acesso em: 30 out. 2016.

GURGEL, Floriano do Amaral; FRANCISCHINI, Paulino G.. **Administração de Materiais e do Patrimônio.** São Paulo: Cengage Learning, 2010. 13 p.

HENRY, Orion; LINDENBAUM, James; WIGGINS, Adam. **Heroku.** Disponível em: <https://www.heroku.com>. Acesso em: 16 ago. 2016.

HEROKU DEV CENTER. **Getting Started with Rails 4.x on Heroku.** Disponível em: <https://devcenter.heroku.com/articles/getting-started-with-rails4>. Acesso em: 16 set. 2016.

KOLODNY, Lora. Medical Technology Dominates at the Olin Cup. **The New York Times.** Nova Iorque. fev. 2010. Disponível em: <http://boss.blogs.nytimes.com/2010/02/07/medical-tech-dominate-at-the-olin-cup/?_r=0#more-10175>. Acesso em: 18 out. 2016.

KULKARNI, Jayant et al. **Quartzy.** Disponível em: <https://www.quartzy.com/about>. Acesso em: 18 out. 2016.

LIMA, Roberto Rodrigues Cunha. LABSTOCKER.COM - PLATAFORMA WEB DE GESTÃO DE REAGENTES QUÍMICOS PARA LABORATÓRIOS. **Congresso Brasileiro de Quimica,** Goiania, nov. 2015. Disponível em: <http://www.abq.org.br/cbq/2015/trabalhos/9/8521-19273.html>. Acesso em: 18 out. 2016.

MATSUMOTO, Yukihiro Matz. **Ruby A PROGRAMMER'S BEST FRIEND.** Disponível em: <https://www.ruby-lang.org/en/>. Acesso em: 26 ago. 2016.

MATSUMOTO, Yukihiro. **Ruby in a Nutshell.** O'reilly, 2001. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.170.3456&rep=rep1&type=pdf>. Acesso em: 14 out. 2016.

NÚCLEO DE TECNOLOGIA DA INFORMAÇÃO (Rio Grande do Sul). Sistema de Controle de Reagentes. Disponível em: <http://www.nti.furg.br/index.php/scr>. Acesso em: 18 out. 2016.

OLBRICH, Kevin C. **Ruby Units.** Disponível em: <https://github.com/olbrich/ruby-units>. Acesso em: 30 out. 2016.

OTTO, Mark et al. **Bootstrap.** Disponível em: <http://getbootstrap.com/>. Acesso em: 28 out. 2016.

PLATAFORMATEC. **Devise.** Disponível em: <https://github.com/plataformatec/devise>. Acesso em: 26 ago. 2016.

PLATAFORMATEC. **Plataformatec.** Disponível em: <http://plataformatec.com.br/>. Acesso em: 26 ago. 2016.

PRESSMAN, Roger S. **Engenharia de Software:** Uma Abordagem Profissional. 7. ed. Porto Alegre: AMGH, 2011. Disponível em:< https://fateczlads.files.wordphttps://fateczlads.files.wordpress.com/2014/08/engenharia-de-software-7c2b0-edic3a7c3a3o-roger-s-pressman-capc3adtulo-1.pdf >. Acesso em: 18 out. 2016.

RAILS. **Imagine what you could build if you learned Ruby on Rails….** Disponível em: <http://rubyonrails.org/>. Acesso em: 14 out. 2016.

REZENDE, Denis Alcides. **Engenharia de Software e Sistemas de Informação.** 3. ed. Rio de Janeiro: Brasport, 2005. Disponível em: <https://books.google.com.br/books?id=rtBvl_L-1mcC&printsec=frontcover&hl=pt-BR&source=gbs_ge_summary_r&cad=0#v=onepage&q&f;=false>. Acesso em: 16 out. 2016.

RUBY ON RAILS HELP. **Setting Up Mailer Using Devise For Forgot Password.** Disponível em: <https://rubyonrailshelp.wordpress.com/2014/01/02/setting-up-mailer-using-devise-for-forgot-password/>. Acesso em: 16 set. 2016.

RUBY, Sam; THOMAS, Dave; HANSSON, David Heinemeier. **Agile Web Development with Rails 4.** 4. ed. Dalas: Rails, 2013. Disponível em: <https://docente.ifrn.edu.br/fellipealeixo/disciplinas/tads-2012/desenvolvimento-de-sistemas-web/book/livro-texto>. Acesso em: 10 ago. 2016.

RUBY. **About Ruby.** Disponível em: <https://www.ruby-lang.org/en/about/>. Acesso em: 14 out. 2016.

SCHWABER, Ken. **Agile Project Management with Scrum.** EUA: Microsoft Press, 2004. 163 p.

SCRUM. **What is a Scrum?.** 2016. Disponível em: <https://www.scrum.org/Resources/What-is-Scrum>. Acesso em: 24 ago. 2016.

SEN, Anith. **Five Simple Database Design Errors You Should Avoid.** Disponível em: <https://www.simple-talk.com/sql/database-administration/five-simple-database-design-errors-you-should-avoid/>. Acesso em: 29 out. 2016.

STEWART, Bruce. **An Interview with the Creator of Ruby.** Disponível em: <http://www.linuxdevcenter.com/pub/a/linux/2001/11/29/ruby.html>. Acesso em: 13 out. 2016.

SUTHERLAND, Jeff; SCHWABER, Ken. **The Scrum Guide.** Disponível em: <http://www.scrumguides.org/scrum-guide.html>. Acesso em: 22 jul. 2016.

TAKEUCHI, Hirotaka; NONAKA, Ikujiro. The new new product development game. **Harvard Business Review,** London, p.137-146, jan. 1986.

TANENBAUM, Andrew S.; VAN STEEN, Maarten. **Distributed Systems: Principles and Paradigms.** 2. ed. Upper Saddle River: Pearson Prentice Hall, 2006. Disponível em: <https://vowi.fsinf.at/images/b/bc/TU_Wien-Verteilte_Systeme_VO_(Göschka)_-_Tannenbaum-distributed_systems_principles_and_paradigms_2nd_edition.pdf>. Acesso em: 16 jul. 2016.

TECHTERMS, **Web Application**. Disponível em:
< http://techterms.com/definition/web_application >. Acesso em: 13 out 2016.

VERHEYEN, Gunther. **Scrum: Framework, not methodology.** 2013. Disponível em: <Scrum: Framework, not methodology>. Acesso em: 22 ago. 2016.

ZAMANI, Hassan. **ActiveRecord::ActsAs.** Disponível em:
<https://github.com/hzamani/active_record-acts_as>. Acesso em: 29 out. 2016.

ANEXO A – Roteiro de Entrevista

1 Contextualização

Explicar o objetivo da entrevista

2 Estabelecendo o perfil do cliente/usuário

Quais são as suas responsabilidades ligadas ao estoque?

Quais são os problemas que interferem no seu trabalho no estoque?

O quê facilita ou dificulta o seu trabalho?

3 Avaliando o Problema

Para cada problema, perguntar:

Por que esse problema existe?

Como resolvê-lo?

4 Entendendo o Ambiente do Usuário

Quais os tipos de usuários/atores do sistema?

Os usuários são experientes nesse tipo de aplicação?

Quais plataformas são usadas?

Outras aplicações usadas são relevantes para essa aplicação? Se sim, fale um pouco sobre elas (ex: sistema de compra de produtos para o estoque).

Quais são as suas expectativas para a usabilidade do produto?

Quais são as suas expectativas para o tempo de treinamento?

Que tipo de auxílio ao usuário você precisa (ex: cópia impressa ou documentos on-line).

5 Recapitulando para Entender

Listar os problemas que o cliente descreveu com as próprias palavras

Eles representam adequadamente os problemas que você está tendo com a solução existente?

Quais outros problemas, caso exista, você está enfrentando?

6 Suposições do Analista sobre o Problema do Cliente

Valide ou invalide suposições

Para cada problema sugerido, pergunte:

Esse é um problema real?

Quais são as razões deste problema?

Como você gostaria de resolvê-lo?

Qual é o peso da solução desse problema, comparado aos outros que você mencionou?

Alguns problemas que já foram pensados previamente:

Dificuldade em padronizar o gerenciamento, já que há muitos tipos diferentes de produtos. Quais tipos são esses?

Dificuldade em saber quando um produto está acabando (uma vez que os produtos são diferentes também)

Dificuldade em saber o status atual do produto - com que está o produto, quantidade, validade.

Dificuldade em fazer o pedido de novos produtos

Importação de dados é um problema?

7 Avaliando a sua solução

Resuma as principais capacidades da solução que você propôs

O que aconteceria se você conseguisse:

Lista de soluções

Como você classificaria cada uma dessas capacidades, por ordem de sua importância?

8 Avaliando a Oportunidade

O quê você considera que seja uma solução bem-sucedida?

9 Avaliando Necessidades de Segurança, Desempenho e Suportabilidade

Quais são os requisitos de segurança?

O que você pensa sobre a manutenção do software?

10 Outros Requisitos

Você acha que existem outros requisitos que devo conhecer?

11 Fechamento

Existe alguma questão que faltou falar?

Você concorda em participar de uma revisão de requisitos em breve?

ANEXO B – Entrevista com Supervisora de Laboratório

Entrevistada: Alcíbia Helena de Azevedo Maia

1 Contextualização

EXPLICAÇÃO SOBRE O OBJETIVO DA ENTREVISTA

2 Estabelecendo o perfil do cliente/usuário

QUAIS SÃO AS SUAS RESPONSABILIDADES LIGADAS AO ESTOQUE?

Responsável principalmente pela parte burocrática, relacionada a UFSC a ao CCS, das compras.

Responsável por fazer memorandos justificando a necessidade de novos materiais

Por exemplo: uma vidraria foi quebrada. A Maitê avisa a Alcíbia e ela faz um memorando, justificando a necessidade da compra da nova vidraria. O memorando passa para o chefe de departamento (Patologia) e depois para o diretor do centro

Responsável por informar quando há a abertura de editais para a compra de materiais. A Alcíbia avisa a Maitê da abertura do edital e a Maitê analisa o quê precisa ser comprado, vai em busca dos orçamentos e finaliza o processo enviando os orçamentos para o financeiro do CCS

Responsável pela orientação da Técnica de Laboratório (Maitê). Ajuda nas tomadas de decisão, uma vez que é a supervisora do Laboratório.

COMO FUNCIONAM OS EDITAIS?

Todo ano existem editais para a compra dos materiais que serão utilizados no ano

Estes editais ocorrem por volta de Maio e são abertos para a UFSC toda

O diretor de centro é o responsável por liberar o dinheiro para cada departamento

QUAIS SÃO OS PROBLEMAS QUE INTERFEREM NO SEU TRABALHO NO ESTOQUE?

Demora na aquisição de materiais. É feita toda a burocracia, mas não há a entrega dos materiais. Os orçamentos são entregues, mas os materiais não chegam.

Estão 3 anos sem receber nada. Não entregam por motivos diversos, o CCS alega que é por culpa da reitoria, a reitoria alega ser culpa do CC

Saber quanto tem de cada material e se estão acabando

QUANTO TEMPO DE ANTECEDÊNCIA SERIA O IDEAL PARA AVISAR DA NECESSIDADE DA COMPRA DE UM NOVO MATERIAL?

Seis meses

COMO É FEITO ATUALMENTE O CONTROLE DO ESTOQUE DE AULA?

Depois de toda aula, antes de entregar as chaves do laboratório, ele precisa completar uma lista de afazeres. Um dos pontos da lista é informar a Maite o quanto de reagente foi utilizado, e se algo está faltando.

O QUÊ FACILITA OU DIFICULTA O SEU TRABALHO?

Às vezes o processo de compras trava no setor financeiro do CCS e não tem continuidade

Existem um monte de ponteiras, mas ninguém sabe quantas têm, e com qual diversidade

VOCÊ MEXE ATUALMENTE NOS ESTOQUES?

Não. Se eu preciso de algo, peço para a Maitê.

3 Avaliando o Problema

PARA CADA PROBLEMA, PERGUNTAR:

POR QUE ESSE PROBLEMA EXISTE?

COMO RESOLVÊ-LO?

Problema 1: Demora na aquisição de materiais

Solução proposta: A Toxicologia ter acesso ao Sistema Solar e poder fazer os pedidos diretamente, sem precisar passar pelo setor financeiro do CCS

Problema 2: Saber quanto tem de cada material e se estão acabando

Solução proposta:

Um sistema de alerta que avise quando um material está acabando - para todos os materiais.

Um sistema de alerta que avise quando um material esteja saindo de validade – apenas para materiais de pesquisa e extensão (não para todos). Mas que seja sempre possível ver a validade para todos os materiais.

Os sistemas de alertas deveriam ser via email e no próprio sistema

Problema 03: Criação de memorandos para compras via CCS

Solução proposta: Selecionar os itens no estoque para gerar um modelo de memorando com os itens marcados. Apenas seria necessário adicionar a justificativa manualmente. Esta solução foi vista como uma “mordomia” pela entrevistada

COMO FUNCIONAM OS MEMORANDOS?

São para pedidos fora dos editais

São necessários uma vez ou outra - são pontuais

Atualmente utiliza-se um modelo de memorando da UFSC

Precisa especificar os materiais que serão pedidos e o por quê dessas necessidades

4 Entendendo o Ambiente do Usuário

QUAIS OS TIPOS DE USUÁRIOS/ATORES DO SISTEMA?

Supervisor de laboratório

Pode cadastrar novos usuários

Pode fazer tudo

Professores

Pode cadastrar monitor

Pode cadastrar responsável pela pesquisa

Pode ver tudo

Técnicos de Laboratório (TAEs)

CRUD de tudo

Se a Toxicologia tiver acesso ao Sistema Solar, este papel teria acesso ao Sistema Solar, pois seria o responsável por inserir os dados lá

Não pode cadastrar novos usuários

Pesquisa

Só CRUD nos materiais de sua responsabilidade

Monitoria

Só CRUD nos materiais de sua responsabilidade

OUTRAS APLICAÇÕES USADAS SÃO RELEVANTES PARA ESSA APLICAÇÃO? SE SIM, FALE UM POUCO SOBRE ELAS.

Sistema Solar:

Foi criado especialmente para Compras e Licitações na UFSC

A Toxicologia tem acesso ao sistema, mas é restrito, apenas de consulta

Para conseguir o acesso de inserção e remoção, precisa pedir autorização ao SETIC

Existem dois códigos para cada material neste sistema

Código da UFSC

Código SIASG (antigo código Catmat) – número de identificação nacional – às vezes pode haver mudança no número

Sistema SIASG:

É um Sistema de Brasília

Responsável por gerar os códigos SIASG

O CCS tem acesso restrito ao SIASG

Se fosse preciso cadastrar um novo material no SIASG, o CCS teria que fazer uma solicitação para os responsáveis adicionarem

Porém, como quase tudo já está cadastrado, não há necessidade de se envolver com este sistema. Tudo é feito através do Sistema Solar na UFSC

Comprasnet

É um portal de compras do governo federal

Caráter mais de consulta

Todos possuem acesso a este sistema

Exemplos de consultas: qual o número SIASG de tal produto? Está ocorrendo algum pregão?...

QUAIS SÃO AS SUAS EXPECTATIVAS PARA O TEMPO DE TREINAMENTO?

Pode ser até uma semana

QUE TIPO DE AUXÍLIO AO USUÁRIO VOCÊ PRECISA? (EX: CÓPIA IMPRESSA OU DOCUMENTOS ON-LINE).

Relatórios de materiais podendo usar filtros

Nome

Categoria e sub: utensílios, reagentes, vidraria, padrões, medicamentos

Validade

Controlados pelo Exército

Controlados pela Polícia Federal

Controlados pela Polícia Civil

O quê está acabando

COMO FUNCIONAM OS PRODUTOS CONTROLADOS DO EXÉRCITO / POLÍCIA CIVIL / POLÍCIA FEDERAL (CONTROLADO VOLUME)?

Existe uma lista de produtos que são controlados por cada um desses órgãos

É necessário mandar uma planilha para cada um dos órgãos, todo mês, informando o quanto foi utilizado de cada material controlado no mês.

São solventes (reagentes líquidos).

5 Recapitulando para Entender

6 Suposições do Analista sobre o Problema do Cliente

7 Avaliando a sua solução

COMO VOCÊ CLASSIFICARIA CADA UMA DESSAS CAPACIDADES, POR ORDEM DE SUA IMPORTÂNCIA?

1 Sistema CRUD para o estoque geral

2 Filtro

2.1 por Nome

2.2 apenas materiais que estão acabando

2.3 por validade

2.4 por estoque (ensino, pesquisa ou geral)

2.5 por categoria

2.6 por subcategoria

2.7 se é controlado por polícia federal, civil ou exército

3. Sistema CRUD para os outros estoques
4. Relatório do filtro
6. Possuir acesso ao Sistema Solar
7. Sistema de papéis usuários
8. Memorando

8 Avaliando a Oportunidade

O QUE VOCÊ CONSIDERA QUE SEJA UMA SOLUÇÃO BEM-SUCEDIDA?

Abrir o sistema e um minuto conseguir saber o estado atual de um material

Conseguir ter uma visão rápida do estoque por meio de filtros para fazer as solicitações anuais em até meio período - sem pensar em orçamento. Apenas saber o quê comprar para um ano. Na tela já seria suficiente, com relatório melhor ainda.

9 Avaliando Necessidades de Segurança, Desempenho e Suportabilidade

10 Outros Requisitos

VOCÊ ACHA QUE EXISTEM OUTROS REQUISITOS QUE DEVO CONHECER?

Não

11 Fechamento

EXISTE ALGUMA QUESTÃO QUE FALTOU FALAR?

Não

VOCÊ CONCORDA EM PARTICIPAR DE UMA REVISÃO DE REQUISITOS EM BREVE?

Sim

ANEXO C – Entrevista com Técnica de Laboratório

Entrevistada: Maitê Perin

1 Contextualização

EXPLICAR O OBJETIVO DA ENTREVISTA

2 Estabelecendo o perfil do cliente/usuário

QUAIS SÃO AS SUAS RESPONSABILIDADES LIGADAS AO ESTOQUE?

Controle de entrada e saída de materiais

Organização do estoque geral

Compra de novos materiais

QUAIS SÃO E COMO FUNCIONAM OS ESTOQUES DO LABORATÓRIO DE TOXICOLOGIA?

Existem 3 estoques: um geral, um para pesquisa e outro para aula/ensino

A Maitê tem responsabilidade pelo estoque geral.

A Maitê entrega ao monitor(a) da disciplina os materiais que ele/ela for precisar, retirando do estoque geral e passando para o estoque de aula. A partir desse momento, os materiais ficam na responsabilidade do monitor(a). Os materiais ficam no estoque de aula até que não sejam mais úteis para a aula, voltando para o estoque geral e para a responsabilidade da Maitê.

O mesmo acontece para pesquisa. Os materiais são entregues para quem estiver realizando pesquisa, retirando do estoque geral e passando para o estoque de pesquisa. A partir desse momento, a responsabilidade sobre os materiais passa para quem os recebeu. Assim que a pesquisa não precisar mais dos materiais, eles são devolvidos para o estoque geral e para a Maite.

A única pessoa que mexe no estoque geral é a Maitê.

COMO ATUALMENTE É FEITO O CONTROLE DE ESTOQUE?

Por meio de planilhas Excel

Apenas a Maite atualiza essas planilhas

O/a monitor(a) e os que estiverem realizando pesquisa informam a Maitê do quanto utilizaram de cada material, e a Maitê atualiza as planilhas

COMO FUNCIONA O PROCESSO DE COMPRAS?

Existem dois tipos de compras, uma direto com o centro e outra com a UFSC

Via UFSC:

Por meio de Licitação

Necessário orçamento de 3 empresas diferentes para cada item a ser comprado

Os valores dos 3 orçamentos para um item devem ser parecidos

Os orçamentos devem possuir validade de 10 dias

O valor de frete deve estar incluso no orçamento

A responsável por conseguir os orçamentos é a Maitê

A Maitê entra em contato com o setor financeiro do CCS, para selecionar no Sistema de Compras e Licitação da UFSC os itens que ela deseja pedir. O sistema gera um modelo de orçamento que deve ser preenchido pelas empresas. O modelo pode ser encontrado nos documentos enviados: Pesquisa_de_Preco – Vidraria.xls

A Maitê envia um email para cada empresa pedindo os orçamentos, com o modelo em anexo. Como as empresas não possuem sempre todos os itens da lista pedida, são necessários vários e-mails para várias empresas.

Após conseguir todos os orçamentos, a Maitê os envia para o setor financeiro do CCS, concretizando o pedido de compra.

O setor financeiro coloca os orçamentos no Sistema de Compras e Licitação da UFSC.

A Maitê não possui acesso ao Sistema de Compras e Licitação da UFSC

Via CCS

Por meio de Memorando

Compra direta

Através do CPF

Utilizando verbas adquiridas via projetos

QUAIS SÃO OS PROBLEMAS QUE INTERFEREM NO SEU TRABALHO NO ESTOQUE?

Saber a quantidade de reagente em cada estoque

Nem sempre o/a monitor(a) ou os que estão realizando pesquisa informam precisamente a Maitê de quanto material foi utilizado.

Às vezes, informam apenas quando o material já acabou. Tendo que conseguir o material as pressas, caso não tenha também no estoque geral.

Controlar materiais que já foram parcialmente utilizados

A embalagem nem sempre está cheia.

Um reagente pode já ter sido utilizado parcialmente, mas não consta essa informação na planilha.

Efetuar compras

Orçamento é a parte mais difícil

Muitos itens

Empresas não respondem com todos os dados de todos os itens

A maioria das empresas possui parte dos itens, não todos

Acaba precisando pesquisar mais de 20 empresas

Demora quase um ano após a entrega dos orçamentos para o financeiro até chegar a mercadoria: pregão, documentação….

COMO É O PROCESSO DE ENTREGA DOS MATERIAIS?

Entregam os materiais na direção de centro. O processo de recebimento é tranquilo

QUÃO FÁCIL É ENCONTRAR OS MATERIAIS NO ESTOQUE?

Com o código de localização é bem fácil encontrá-los

3 Avaliando o Problema

PARA CADA PROBLEMA, PERGUNTAR:

POR QUE ESSE PROBLEMA EXISTE?

COMO RESOLVÊ-LO?

Problema 1: Saber a quantidade de reagente em cada estoque

Solução proposta:

Criar uma planilha de uso para aula e outra para pesquisa

Responsável pela pesquisa e pela aula deveriam anotar diariamente a quantidade dos produtos que estão sendo utilizados.

A prioridade desta planilha seria para aula

Registrar também os descartes na planilha. Uma vez que, sempre que vão ao laboratório para recolher os resíduos, é preciso informar o quê consta nos resíduos, bem como as quantidades

Todo fim do semestre, o monitor de aula mostraria um relatório informando o quê será necessário para o próximo semestre.

Problema 2: Controlar materiais que já foram parcialmente utilizados
Solução proposta: Adicionar algum tipo de marcação no sistema para indicar se o material já foi utilizado ou não

Problema 3: Efetuar compras
Solução proposta:

Criar um modelo de orçamento para enviar para as empresas

Selecionar os itens que devem ser comprados no sistema de estoque, e a partir desses dados, gerar um modelo de orçamento a ser preenchido, enviando os dados por email para as empresas.

O Sistema Solar já faz isso, porém nem toda compra é feita através da UFSC, ou seja, nem toda compra é feita utilizando o Sistema Solar. Para as compras diretas e via CCS não tem como gerar este modelo automaticamente ainda.

4 Entendendo o Ambiente do Usuário

QUAIS OS TIPOS DE USUÁRIOS/ATORES DO SISTEMA?

Admin

Possui responsabilidade e acesso (CRUD) sobre todos os produtos

Pesquisador

Apenas uma pessoa responsável pela pesquisa

Só possui acesso (CRUD) aos materiais que possui responsabilidade sobre - os que foram entregues do estoque geral pela Maitê

Monitor

Apenas uma pessoa responsável pela aula – monitor(a)

Só possui acesso (CRUD) aos materiais que possui responsabilidade sobre - os que foram entregues do estoque geral pela Maitê

Sempre deve ser possível trocar a responsabilidade de um produto

OS USUÁRIOS SÃO EXPERIENTES NESSE TIPO DE APLICAÇÃO?

Sim

QUAIS PLATAFORMAS SÃO USADAS?

Excel

OUTRAS APLICAÇÕES USADAS SÃO RELEVANTES PARA ESSA APLICAÇÃO? SE SIM, FALE UM POUCO SOBRE ELAS.

O Sistema Solar – que contém vários outros subsistemas, inclusive o Sistema de Compras e Licitação, o sistema financeiro...

QUAIS SÃO AS SUAS EXPECTATIVAS PARA A USABILIDADE DO PRODUTO?

Que o layout não seja o “padrão” da UFSC, por não ser intuitivo

QUAIS SÃO AS SUAS EXPECTATIVAS PARA O TEMPO DE TREINAMENTO?

30 minutos seriam o suficiente.

QUE TIPO DE AUXÍLIO AO USUÁRIO VOCÊ PRECISA? (EX: CÓPIA IMPRESSA OU DOCUMENTOS ON-LINE).

O modelo de documento para orçamento, que já foi citado anteriormente.

5 Recapitulando para Entender

QUAIS OUTROS PROBLEMAS, CASO EXISTA, VOCÊ ESTÁ ENFRENTANDO?

Problema 4: Saber a quantidade de ponteiras presente no estoque. Por muitas vezes não se tem noção da quantidade de ponteiras que existe no estoque. Existe um saco com uma infinidade de ponteiras, mas ninguém se dá o trabalho de contar quantas, pois são muitas.

Solução proposta:

Adicionar algum tipo de marcação no sistema, análogo ao proposto no problema 2, para indicar a fração de ponteiras que ainda existe no saco.

COMO SÃO CLASSIFICADOS OS MATERIAIS NO ESTOQUE, TANTO PARA ORGANIZAÇÃO COMO QUANTO PARA ORÇAMENTO? O QUÊ É IMPORTANTE SABER SOBRE CADA CLASSIFICAÇÃO?

Categoria 1: Utensílios

Informações relevantes:

Código da UFSC

Descrição

Quantidade: por unidade

Exemplos: estante de tubo, espátula...

Categoria 2: Vidrarias

Informações relevantes:

Código da UFSC

Descrição

Quantidade: por unidade

Volume (Ainda precisam ser contabilizadas no estoque)

Categoria 3: Reagentes

Informações relevantes:

Código da UFSC

CAS (nº com registro único no banco de dados do Chemical Abstracts Service)

Descrição

Grau de pureza (para pesquisa precisa ser altamente puro, já na aula não é necessário

Sinônimo

Marca

Validade (caso for importado, informar que não possui este campo)

Estado de conservação: ruim, médio, bom

Subcategoria: Líquidos

Informações relevantes:

Quantidade: sempre em ml

Obs: No contexto do estoque, não faz diferença se é corrosivo, inflamável…

Subcategoria: Sólidos

Informações relevantes:

Quantidade: sempre em gramas

Formato: barra, pó ou pastilha

Categoria 4: Padrões

Informações relevantes:

Descrição

Concentração

Volume (ml ou g)

Validade

Podem ser sólidos ou líquidos

Devem ser novos e em bom estado

Para a análise de cocaína, precisa-se de uma solução em que é de conhecimento a presença de cocaína, bem como a sua concentração, para efetuar as análises.

As quantidades recebidas são pequenas

Muitas vezes a compra é feita diretamente com o IGP

Ainda não ocorre a tramitação legal para drogas, mas existe

Ficam na geladeira de pesquisa ou de aula, e não nos estoques.

Quem cuida são os responsáveis da aula e da monitoria, estoque geral não interfere

Categoria 5: Medicamentos

Não estão sendo muito utilizados no momento

Ainda não sabem se vão querer controlar no estoque

A VALIDADE DOS MATERIAIS É IMPORTANTE?

Produtos nacionais tem validades, mas internacionais não

Seria interessante que o sistema marcasse que um produto é importado e por isso não tem validade

A validade é importante para a pesquisa, já não tanto para as aulas

Não se pode utilizar para pesquisa um produto vencido, uma vez que isso pode alterar nos resultados

Já para as aulas, existem produtos fora da validade que funcionam muito bem ainda

O quê é mais importante é o estado de conservação do material: ruim, médio, bom

6 Suposições do Analista sobre o Problema do Cliente

7 Avaliando a sua solução

COMO VOCÊ CLASSIFICARIA CADA UMA DESSAS CAPACIDADES, POR ORDEM DE SUA IMPORTÂNCIA?

1. Ter controle do estoque geral (apesar de já estar sendo feito via Excel)
    1.1 Reagentes: Líquido e Sólido
    1.2 Padrões
    1.3 Utensílio
1.4 Vidraria
    1.5 Medicamento

2. Saber a quantidade de reagente no estoque de aula e de pesquisa
3. Saber se o material já foi parcialmente utilizado
4. Saber sobre o estado de conservação do material
5. Modelo de orçamento e envio automaticamente de email para as empresas

8 Avaliando a Oportunidade

O QUE VOCÊ CONSIDERA QUE SEJA UMA SOLUÇÃO BEM-SUCEDIDA?

Em uma manhã conseguir fazer o controle de uma categoria.

9 Avaliando Necessidades de Segurança, Desempenho e Suportabilidade

QUAIS SÃO OS REQUISITOS DE SEGURANÇA?

Que os usuários apenas possam alterar os produtos na sua responsabilidade

Que o admin tenha responsabilidade sobre todos os produtos

E que seja possível trocar a responsabilidade de um produto sempre

10 Outros Requisitos

VOCÊ ACHA QUE EXISTEM OUTROS REQUISITOS QUE DEVO CONHECER?

Não

11 Fechamento

EXISTE ALGUMA QUESTÃO QUE FALTOU FALAR?

Não

VOCÊ CONCORDA EM PARTICIPAR DE UMA REVISÃO DE REQUISITOS EM BREVE?

Sim

ANEXO D – Instruções de Instalação e Implantação

MyLab

Web Application for Stock Management for the laboratory "Laboratório de Pesquisas Toxicológicas" (LPTox) from the university "Universidade Federal de Santa Catarina"

Website: https://agile-springs-54168.herokuapp.com/

Deploy: Localhost

Requirements

- Ruby - This project uses `.ruby-version`, `.ruby-gemset` and `Gemfile` to specify dependencies. Make sure that `rvm` or `rbenv` is installed on your computer. Check `rvm info` command to see that the correct ruby version and gemset are set. Also make sure that `./bin` is in your `$PATH` in the first place.
- Rails 4
- PostgreSQL 9.4

Repositories Setup

```
git clone git@github.com:camilamaia/mylab.git
cd mylab
bundle
bundle update
```

Postgres Setup

```
sudo -u postgres psql
postgres=# CREATE USER mylab WITH PASSWORD 'lab';
postgres=# ALTER USER mylab CREATEDB;
```

From mylab directory:

```
rake db:create
rake db:migrate
```

Rails Run

From mylab directory:

```
rails s
```

Browser

To see the application running, you only need to open a web browser on localhost and port 3000:

```
firefox http://localhost:3000
```

Deploy: Production Heroku

From mylab directory, with your changes already pushed to git:

```
$ git push heroku master
```

or, for a specific branch:

```
$ git push heroku yourbranch:master
```

And, migrate it:

```
$ heroku run rake db:migrate
```

UFSC Webserver Configuration

Connect to UFSC VPN using your UFSC username and password

```
server: srv1.vpn.ufsc.br
```

Access the server using your server username and password

```
ssh <username>@150.162.6.107
```

Requirements

- Docker requires a 64-bit installation regardless of Debian version. Additionally, the kernel must be 3.10 at minimum. The latest 3.10 minor version or a newer maintained version are also acceptable. Kernels older than 3.10 lack some of the features required to run Docker containers. checking the kernel version:

```
$ uname -r
4.4.0-15-generic
```

Update your apt repository

Docker's APT repository contains Docker 1.7.1 and higher. To set APT to use from the new repository:

Update package information, ensure that APT works with the https method, and that CA certificates are installed

```
$ apt-get update
$ apt-get install apt-transport-https ca-certificates
```

Add the new GPG key

```
$ apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys
58118E89F3A912897C070ADBF76221572C52609D
```

Open the */etc/apt/sources.list.d/docker.list* file (if the file doesn't exist, create it).

Remove any existing entries

Add an entry for your Debian operating system. The possible entries are (On Debian Stretch/Sid):

```
deb https://apt.dockerproject.org/repo debian-stretch main
```

Update apt-get

```
$ apt-get update
```

Verify that APT is pulling from the right repository

```
$ apt-cache policy docker-engine
```

From now on when you run apt-get upgrade, APT pulls from the new apt repository.

Install Docker

Before installing Docker, make sure you have set your APT repository correctly as described in the prerequisites.

Update the APT package index.

```
 $ sudo apt-get update
```

Install Docker.

```
$ sudo apt-get install docker-engine
```

Start the docker daemon.

```
 $ sudo service docker start
```

Verify docker is installed correctly.

```
 $ sudo docker run hello-world
```

This command downloads a test image and runs it in a container. When the container runs, it prints an informational message. Then, it exits.

Give non-root access

The docker daemon always runs as the root user and the docker daemon binds to a Unix socket instead of a TCP port. By default that Unix socket is owned by the user root, and so, by default, you can access it with sudo.

If you (or your Docker installer) create a Unix group called docker and add users to it, then the dockerdaemon will make the ownership of the Unix socket read/writable by the docker group when the daemon starts.

The docker daemon must always run as the root user, but if you run the docker client as a user in the docker group then you don’t need to add sudo to all the client commands. From Docker 0.9.0 you can use the -G flag to specify an alternative group.

Add the docker group if it doesn't already exist.

```
$ sudo groupadd docker
$ sudo gpasswd -a <username> docker
```

Restart the Docker daemon.

```
$ sudo service docker restart
```

Install Docker Compose

```
$ sudo curl -L
https://github.com/docker/compose/releases/download/1.8.0/docker-compose-`uname
-s`-`uname -m` > /usr/local/bin/docker-compose
$ sudo chmod +x /usr/local/bin/docker-compose
```

Test the installation

```
$ docker-compose --version
docker-compose version: 1.8.1
```

Project Configuration

Clone the application repository

```
$ git clone git@github.com:camilamaia/mylab.git
$ cd mylab
```

Start by setting up the two files you'll need to build the app. First, since your app is going to run inside a Docker container containing all of its dependencies, you'll need to define exactly what needs to be included in the container. This is done using a file called Dockerfile. To begin with, the Dockerfile consists of:

```
$ vim Dockerfile
```

```
FROM ruby:2.2.2 # this specifies an existing container with ruby 2.2.2 in the
docker cloud

RUN apt-get update -qq # run commands inside ruby container
RUN apt-get install -y build-essential libpq-dev locales apt-utils debconf-utils
openssl libssl-dev vim
ENV LANG=C.UTF-8 LANGUAGE=en_US:en LC_ALL=C.UTF-8

RUN mkdir /mylab
WORKDIR /mylab # set basedir for the next commands
ADD Gemfile /mylab/Gemfile # add the host Gemfile into the container
ADD Gemfile.lock /mylab/Gemfile.lock

RUN bundle install

ADD . /mylab #add everything on the local host at the current directory into the
container

ENV RAILS_ENV production

EXPOSE 3000 # exposes 3000 port of the container to the host, can be commented
#because we will define this again on the compose file

ENV HOSTNAME mylab.ufsc.br
ENV DOMAIN ufsc.br
```

That'll put your application code inside an image that will build a container with Ruby, Bundler and all your dependencies inside it.

Finally, docker-compose.yml is where the magic happens. This file describes the services that comprise your app (a database and a web app), how to get each one's Docker image, and the configuration needed to link them together and expose the web app's port.

```
$ vim docker-compose.yml
```

```
version: '2'
services:
  postgres: # postgres service description
    image: postgres # image used by the container (official postgres' image)
    volumes: # define the volume that the container will share with the hostmachine
      -  postgresql_data:/var/lib/postgresql/data
    hostname: postgres
    environment:
      - "PRODUCTION_PASSWORD=postgres"
      - "PRODUCTION_USERNAME=postgres"
      - "PRODUCTION_DATABASE=mylab_production"
 #   restart: always #in case the database dies
###    ports:
#      - "15432:5432" # links the 5432 port of the host on the 5432 port of the
container
#to make database accessible from the host machine. It can be commented because
#the container already exposes the 5432 port automatically which is specified on
the
#Dockerfile of the official postgres image and we don't need an exposed port on the
#host machine for the database
  rails: #rails service description
    build: . # this means that the build is refered in a Dockerfile in the current
directory
that's why we don't need to set a image here, the Dockerfile already have a rails
image
    command: bundle exec rails s -p 3000 -b '0.0.0.0' -e production # the command
executed for this service
    links:
      - postgres # rails service connects to the postgres service ( rails can
LISTEN on postgres port)
    environment:
      - "RAILS_ENV=production"
      - "DOMAIN=ufsc.br"
      - "HOSTNAME=mylab.ufsc.br"
    ports: # rails exposes the port 3000 and the host machine make this service
available on 80 port
      - "80:3000"
volumes:
  postgresql_data:
# restart: always # in case the service dies
```

Run the build parts of the compose file downloading the necessary images and creating the containers

```
$ docker-compose build
```

```
```

The containers will run along with the services inside and detaching from the console output

```
 $ docker-compose up -d
```

Check the logs till the rails server is up and running

```
$ docker-compose logs
```

Run rake db:create on rails container

```
$ docker-compose run rails rake db:create
```

Check the logs of the rake tasks in both containers

```
$ docker-compose logs
```

Run rake migrate on rails container

```
$ docker-compose run rails rake db:migrate
```

Check if everything went up nicely

```
$ docker-compose logs
```

Now just check the browser

```
http://mylab.ufsc.br
```

Commit the Dockerfile and the docker-compose.yml into the repo.

Deploy: Production UFSC Webserver

Connect to UFSC VPN using your UFSC username and password

```
 server: srv1.vpn.ufsc.br
```

Access the server using your server username and password

```
ssh <username>@150.162.6.107
```

Acess the project folder

```
$ cd /home/camilamaia/mylab
```

Pull the code changes

```
$ git pull origin master
```

Code changes should be added to the container so we must remove the old container that contains an earlier version of the code

Stop all containers associated to compose file

```
$ docker-compose stop
```

Show the containers list

```
$ docker ps -a -q
```

Remove all containers

```
$ docker rm $(docker ps -a -q)
```

Remove the images (only necessary when Dockfile has changes)

```
$ docker rmi mylab_rails
```

Show volume info

```
$ docker volume inspect mylab_postgresql_data
 {
        "Name": "mylab_postgresql_data",
        "Driver": "local",
        "Mountpoint": "/var/lib/docker/volumes/mylab_postgresql_data/_data",
        "Labels": null,
        "Scope": "local"
    }
```

The value MountPoint shows where the data is stored on the host machine. So, before you rebuild

the containers with the changes made to the code, make a backup of this volume:

```
$ sudo cp -a /var/lib/docker/volumes/mylab_postgresql_data/_data
/home/camilamaia/mylab_backup/_data_OLD
```

(If rollback is needed, replace the _data folder content with the _data_OLD content)

Check that the copy was made properly

```
$ ls /home/camilamaia/mylab_backup/
```

Now, let's rebuild the containers

```
$ docker-compose build
$ docker-compose up -d
$ docker-compose run rails rake db:migrate
```

And, check the changes on the browser

```
http://mylab.ufsc.br
```

ANEXO E – Instruções para Acrescentar Nova Categoria

Add New Category

Create new category

Generate the scaffold (Don't need to add name and description - they are already on Material model)

```
$ cd mylab
$ rails generate scaffold <CategoryName> attribute1:type1 attribute2:type2
--no-timestamps
```

example:

```
$ rails generate scaffold Glassware ufsc_id:integer capacity:string
quantity:integer --no-timestamps
```

Migrate the database

```
rake db:migrate
```

Edit model file app/models/\<category_name>.rb. Copy from app/models/glassware.rb and change the validations and the class declaration properly

Edit controller file app/controllers/\<category_name>_controller.rb. Copy from app/controllers/glassware_controllers.rb. Edit the variable names properly according the category name. Update the permit params with name, description and the others attributes of the category.

Edit description. Edit app/models/material.rb file, method attr_friendly_description - update if a new attribut added

Edit view files and delete the useless too

```
app/views/<category_name>/_form.html.slim
  Copy from app/views/glasswares/_form.html.slim
  Change the variable names and fields
app/views/<category_name>/edit.html.slim
  Copy from app/views/glasswares/edit.html.slim
  Change the variable names
app/views/<category_name>/new.html.slim
  Copy from app/views/glasswares/new.html.slim
  Change the variable names
app/views/<category_name>/show.html.slim (edit fields too)
  Copy from app/views/glasswares>/show.html.slim
  Change the variable names and fields
```

Add category on materials view dropdown: app/views/materials/index.html.slim

Add new locales for the new category and new fields

```
config/locales/material/en.yml
config/locales/material/pt-BR.yml
config/locales/material_category/en.yml
config/locales/material_category/pt-BR.yml
config/locales/simple_form.pt-BR.yml
config/locales/simple_form.en.yml
```

Create new category item

Create item category scaffold

```
$ cd mylab
$ rails generate scaffold Item<CategoryName> attribute1:type1 attribute2:type2
--no-timestamps
```

example:

```
$ rails generate scaffold ItemGlassware current_quantity:string --no-timestamps
```

Migrate the database

```
rake db:migrate
```

Edit model file app/models/item_\<category_name>.rb. Copy from app/models/item_glassware.rb and change the validations and the class declaration properly

Edit controller file app/controllers/item_\<category_name>_controller.rb. Copy from app/controllers/item_glassware_controllers.rb. Edit the variable names properly according the category name. Update the permit params with :material_id, :name and the others attributes of the category.

Edit description. Edit app/models/item.rb file, method attr_friendly_description - update if a new attribut added

Edit views file and delete the useless too.

```
* app/views/item_<category_name>/_form.html.slim
  Copy from app/views/item_glasswares/_form.html.slim
  Change the variable names and fields
* app/views/item_<category_name>/edit.html.slim
  Copy from app/views/item_glasswares/edit.html.slim
  Change the variable names
* app/views/item_<category_name>/new.html.slim
  Copy from app/views/item_glasswares/new.html.slim
```

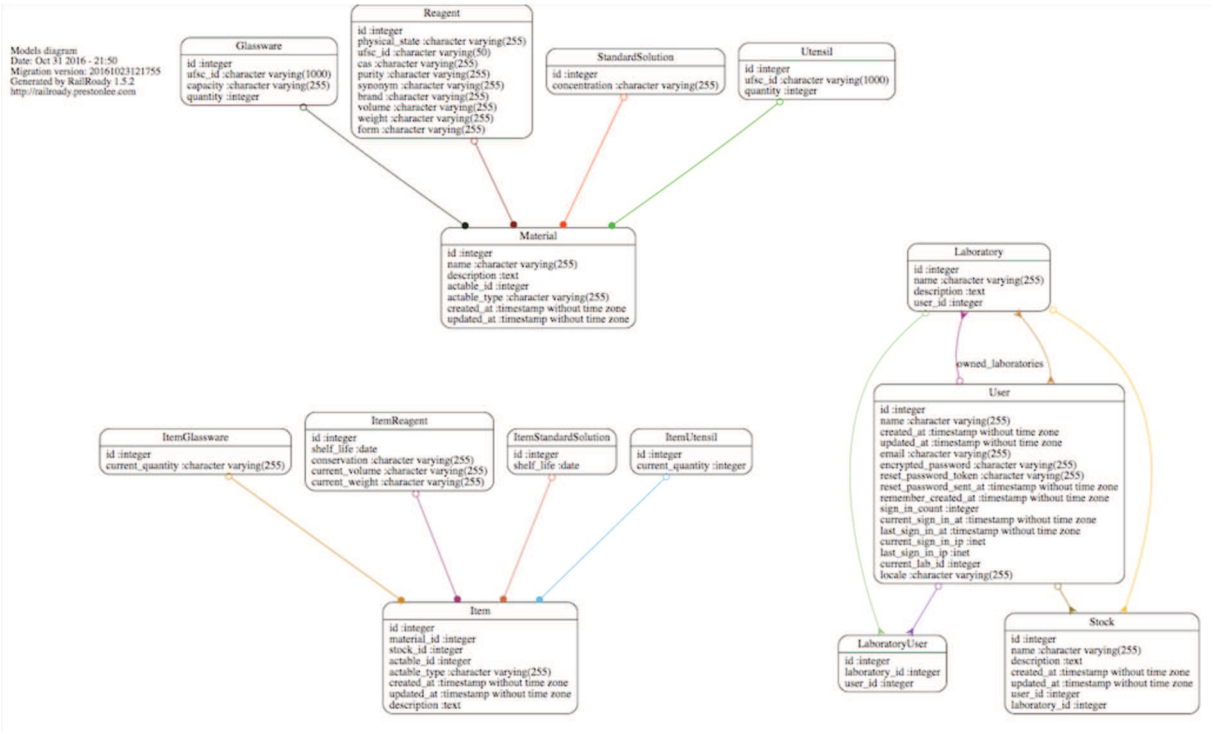
```
  Change the variable names
* app/views/item_<category_name>/show.html.slim (edit fields too)
  Copy from app/views/item_glasswares>/show.html.slim
  Change the variable names and fields
```

Add item category on items view dropdown: app/views/items/index.html.slim

Add new locales for the new category and new fields

```
config/locales/item/en.yml
config/locales/item/pt-BR.yml
config/locales/item_category/en.yml
config/locales/item_category/pt-BR
config/locales/simple_form.pt-BR.yml
config/locales/simple_form.en.yml
```

ANEXO F – Diagramas de classes: Modelos



APÊNDICE A

Aplicação Web para Controle de Estoques do Laboratório de Pesquisas Toxicológicas da Universidade Federal de Santa Catarina

Camila M. Maia[1], Leandro José Komosinski[1]

[1]Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC) – Florianópolis, SC – Brasil

camilamaia@inf.ufsc.br, leandro.komosinski@ufsc.br

Abstract. *Manage inventory is a known issue for a long time. It is really difficult to ensure that products are available and to avoid waste. This article proposes a web application to assist the LPTox/UFSC inventory control. The software optimizes inventory management, in addition to the record the stored items in a clear and organized way, also sends alerts when supplies are finishing or are about to be out of shelf life; it allows searches of materials through filters, allowing different views of the whole picture; and generates documents that facilitate the order of new materials. The prototype is still in the development phase.*

Resumo. *Gerenciar estoques é um problema conhecido há bastante tempo na administração. Garantir que os produtos necessários estejam à disposição e evitar perdas é uma balança difícil de equilibrar. Este artigo apresenta como solução uma aplicação web que auxilia no controle de estoque do LPTox. A ferramenta otimiza a gerência de estoque, além do registro dos itens armazenados de maneira clara e organizada, também emite alertas de quando materiais estão por acabar ou estão prestes a sair da validade; permite buscas de materiais por meio de filtros, possibilitando visões diferentes do todo; e gera documentos que facilitam o pedido de novos materiais. O protótipo encontra-se ainda em fase de desenvolvimento.*

1. Introdução

Laboratórios de química em geral possuem muitos reagentes, vidrarias, utensílios e outros itens que precisam ser armazenados e controlados apropriadamente. O controle de estoque é uma área da administração que consiste em gerenciar tudo o que envolve cada item de um estoque, desde fornecedores, pedidos, localização dos itens, quantidades, validades e o quê mais for necessário para o controle pleno desta área.

Entretanto, gerenciar um estoque nem sempre é uma tarefa fácil. Quando não bem executada, pode causar sérios prejuízos ao laboratório, como perda de dinheiro, ausência de materiais para usos primordiais, materiais fora da validade e claro, muita perda de tempo ao ter que trabalhar com este caos.

O Laboratório de Pesquisas Toxicológicas (LPTox) da Universidade Federal de Santa Catarina (UFSC) possui um vasto estoque e se enquadra nesta situação caótica

citada anteriormente. Atualmente, o laboratório tenta desenvolver seu próprio método de controle de estoque, porém, de maneira falha e ineficiente. Materiais acabam, produtos estão fora da validade e nada disso é percebido até se deparar com o problema. Aliado a isso, existe ainda a grande perda de tempo no processo de novos pedidos de materiais.

O gerenciamento de estoques de laboratório é notoriamente imprescindível para um bom funcionamento do laboratório. Pode-se avaliar a grandiosidade do problema analisando os softwares já existentes no mercado. A aplicação Quartzy, por exemplo, é utilizado por mais de 30.000 laboratórios ao redor do mundo (KULKARNI et al., 2016), reforçando que o problema é sim real. Contudo, as necessidades do LPTox são um pouco mais específicas e os programas existentes no mercado analisados não eram totalmente adequados para a tarefa.

A metodologia utilizada para o desenvolvimento da aplicação foi baseada no framework Scrum, visando um desenvolvimento ágil e eficaz. O protótipo se encontra ainda em fase de desenvolvimento. Em entrevista com a supervisora do LPTox, ela afirma que, com as funcionalidades atuais que o software apresenta, tem-se 70% dos casos de uso cobertos. Com a implementação do filtro de materiais, esse número já atingiria a casa dos 90%.

todo

2. Abordagem metodológica

2.1. Adaptação do Scrum para Um Único Membro

A metodologia de desenvolvimento de software abordada foi uma adaptação do Scrum, um *framework* de desenvolvimento ágil de *software* iterativo e incremental para gerenciar o desenvolvimento de produtos (SCRUM, 2016) (VERHEYEN, 2013). A adaptação foi necessária uma vez que existe somente um membro no projeto, e não uma equipe, como previsto no Scrum.

Com a existência de um único membro, os três papéis são atribuídos para a mesma pessoa, ou seja, a mesma pessoa cumpre as funções de Product Owner, de Scrum Master e de time de desenvolvimento. O evento de Daily Scrum passa a ser desnecessário neste contexto, porém todos os outros eventos são mantidos, com a única alteração de que não são mais reuniões, mas sim avaliações individuais sobre cada aspecto. Os artefatos também se mantêm inalterados.

2.2. Estratégia para Levantamento de Requisitos

O levantamento de requisitos, ou levantamento de dados, é o processo de reunir informações sobre os sistemas existentes e sobre o sistema proposto (FIRPO, 2011). Para o levantamento de requisitos desta aplicação foram utilizadas três técnicas diferentes: entrevista, análise de documentos e prototipação, visando garantir o melhor aproveitamento possível.

A entrevista foi a primeira técnica a ser aplicada. As entrevistas possibilitaram a criação dos requisitos funcionais do sistema e uma maior compreensão da aplicação. Foi solicitado que as participantes entrevistados enviassem todo o tipo de documento que

pudesse auxiliar na compreensão do contexto. Foram enviados um total de nove documentos que auxiliaram, principalmente, sanando dúvidas referentes aos requisitos.

A última técnica aplicada foi a prototipação, um processo iterativo durante todo o desenvolvimento do software. As sessões consistiam em deixar o usuário livre para o uso da aplicação, sem orientar como esta deveria ser utilizada. Desta maneira, foi possível verificar a usabilidade do software, bem como se os requisitos estavam sendo implementados como o desejado. Ao final de todas as sessões, os requisitos eram atualizados, se necessário, e as modificações necessárias na aplicação eram planejadas para seguinte etapa de desenvolvimento.

Aplicando as três técnicas, foi possível construir um conjunto de requisitos sólido de agrado ao usuário, assim como uma aplicação que reflete tais requisitos da melhor forma possível – uma aplicação bem-sucedida - sempre visando os desejos do usuário final.

A figura 1 mostra o diagrama de casos de uso construído a partir do levantamento de requisitos.

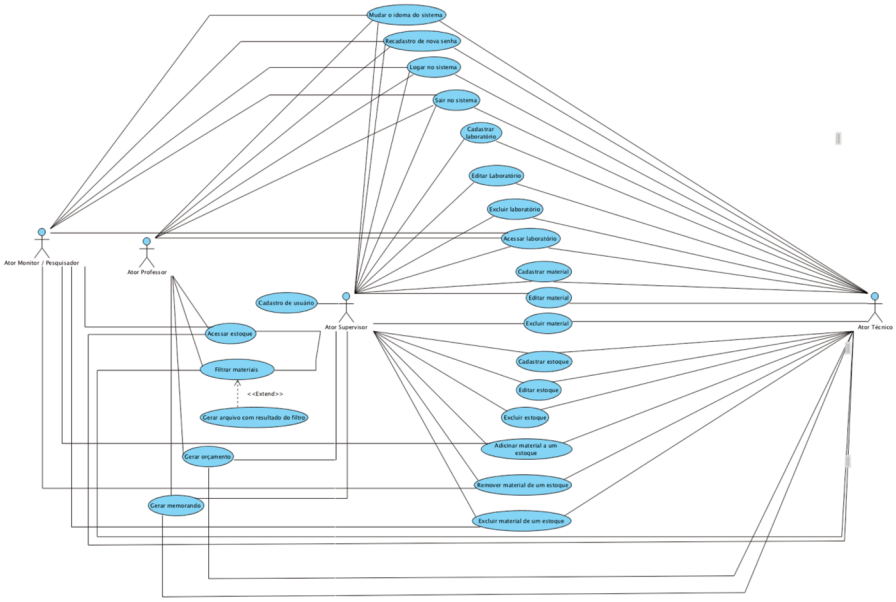


Figure 1. Diagrama de Casos de Uso

3. Arquitetura da Solução

A aplicação em questão foi construída utilizando o framework Ruby on Rails 4 sendo assim, sua arquitetura segue fielmente a arquitetura deste framework. Como pede o *framework* Rails, foi utilizado o padrão de projeto *Model-View-Controller* (MVC) para implementação da Camada de Aplicação.

Aplicando o MVC para o Rails, quando um usuário gera alguma requisição através do navegador, essa requisição é primeiramente enviada ao roteador, que é responsável por analisá-la e repassá-la para o controlador apropriado. O roteador deve também indicar qual método deste controlador será incumbido de receber a requisição - esse tipo de método é conhecido como uma ação do controlador. A ação tem acesso aos dados da requisição, pode interagir com o modelo, pode chamar outras ações e pode, eventualmente, preparar informações para a visão, que exibe algo para o usuário no navegador.

A figura 2 refere-se a um exemplo dado pelo livro "Agile Web Development with Rails 4" (RUBY et al., 2013) que ilustra precisamente como a arquitetura Rails funciona e é composta.

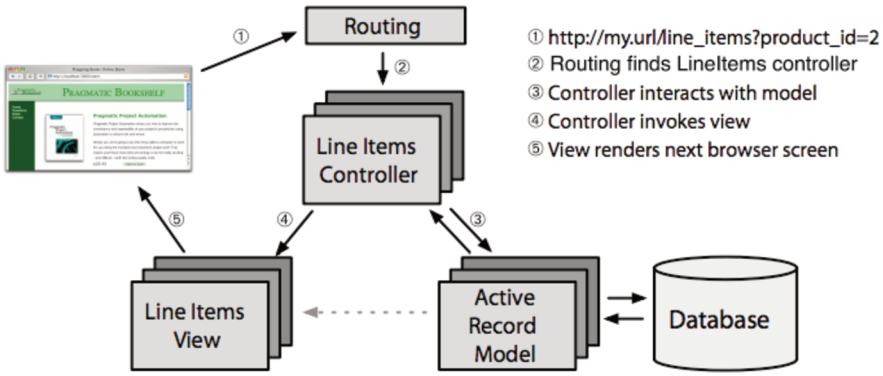


Figura 2. Rails e MVC

Dessa forma, seguir a arquitetura do Rails proporciona agilidade, facilidade de desenvolvimento, sendo o Scaffold um exemplo disso, e alta manutenibilidade, uma vez que implementa o MVC.

4. Desenvolvimento

4.1. Tecnologias Utilizadas

Para o desenvolvimento da aplicação foi utilizada a linguagem de programação Ruby e os *frameworks* Ruby On Rails e Bootstrap.

Ruby é uma linguagem de programação interpretada criada pelo Japonês Yukihiro "Matz" Matsumoto e lançada publicamente no ano de 1995 (RUBY, 2016). Escrita em C, segundo David Flanagan e Yukihiro Matsumoto no livro The Ruby Programming Language (FLANAGAN; MATSUMOTO, 2008) "Ruby inspira-se em Lisp, Smalltalk e Perl, mas usa uma gramática que é de fácil aprendizado para programadores C e Java™. Pela proposta de alta velocidade de desenvolvimento, por ser orientado a objetos, por ser flexível, aceitando também programação funcional, por ser open source, e por possuir um framework, Ruby On Rails (tema de próxima seção), que proporciona um conjunto de ferramentas muito vasto para aplicações web, Ruby foi escolhida como a linguagem de programação para o desenvolvimento desta aplicação.

Ruby on Rails, ou somente Rails, é um *framework* de desenvolvimento de aplicações web escritas na linguagem Ruby. Ele é projetado para tornar a programação de aplicações web mais fácil, fazendo suposições sobre o quê cada desenvolvedor precisa para começar (BIGG et al., 2016). Por ser um framework open source, que promete aumentar velocidade e facilidade no desenvolvimento de sites orientados a banco de dados, exatamente o caso de uma aplicação de gerenciamento de estoque, optou-se por utilizar Ruby on Rails neste projeto.

O Bootstrap fornece definições básicas de estilo para todos os componentes chave HTML por meio de um vasto conjunto de folhas de estilo. Além disso, também possui elementos JavaScript na forma de *plugins* jQuery. Optou-se por utilizar Bootstrap nesse projeto por ser open source, proporcionar design responsivo, possuir ótimo sistema de grid, por tornar o desenvolvimento da visão da aplicação muito mais ágil, por possibilitar a reutilização de componentes, fornecendo maior manutenibilidade e principalmente, por facilitar a criação de interfaces gráficas uniformes, consistes e modernas.

4.2. Implementação

A aplicação foi desenvolvida em quarto *Sprints* de uma semana cada. *Sprint* 1: Estoque e Usuário; Sprint 2: Laboratório e Ambiente de Produção; Sprint 3: Material e Internacionalização; Sprint 4: Material em Estoque e Domínio da UFSC. Ao final do desenvolvimento da Sprint 4, o protótipo foi apresentado conforme as figuras 3 e 4.

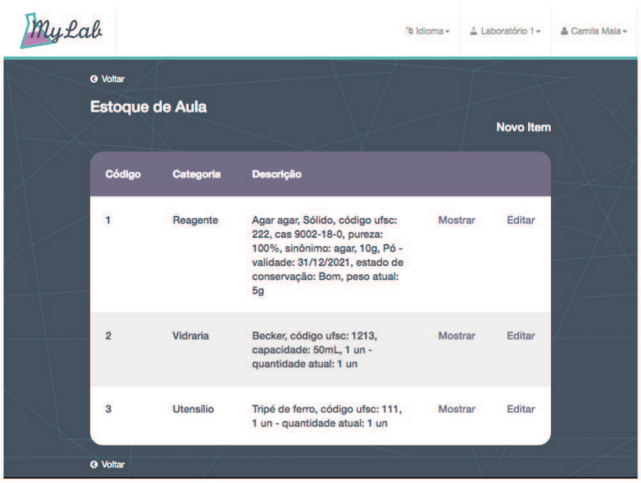


Figura 3. Tela de acesso ao estoque.

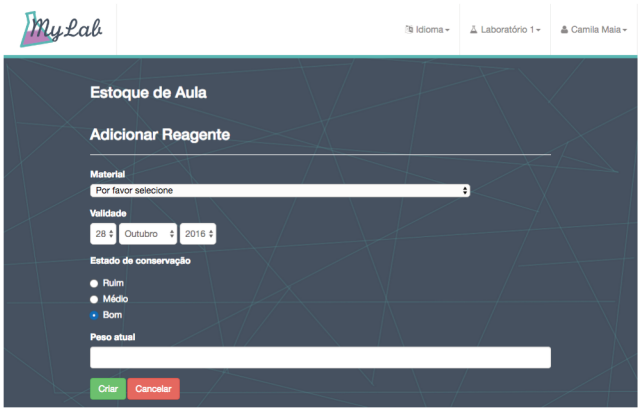


Figura 4. Tela de novo item de reagente sólido.

Com a implantação da aplicação tanto no servidor do Heroku como no servidor da UFSC, o protótipo pode ser testado pela supervisora e pela técnica de laboratório do LPTox ao longo das etapas de desenvolvimento.

Mesmo sendo ainda um protótipo, em entrevista com a supervisora do LPTox, ela afirma que, com as funcionalidades atuais que o software apresenta, tem-se 70% dos casos de uso cobertos. Com a implementação do filtro de materiais, esse número já atingiria a casa dos 90%.

5. Conclusões

Conclui-se que o gerenciamento de estoque é um problema recorrente e importante para laboratórios de química. Que a ausência de um controle adequado de estoque pode levar a perda de dinheiro, de materiais e de tempo. Conclui-se também que aplicações webs podem sim ajudar a resolver esse tipo de situação. Inclusive, que já existem aplicações presentes no mundo acadêmico e no mercado com esse objetivo. Por outro lado, tais soluções não atendem a todas as necessidades específicas que o LPTox apresentou, sendo então de relevância o desenvolvimento do presente projeto.

O levantamento de requisitos utilizando as três técnicas: entrevistas, análise de documentação e prototipação foi visto como fundamental para o fornecimento de conhecimento e embasamento para a criação de um conjunto de requisitos sólidos, que foram sendo adaptados ao longo de todo o processo de desenvolvimento do trabalho. Analisou-se, também, que a metodologia adotada, o framework Scrum, proporcionou como benefícios a iteratividade, organização, velocidade, e garantiu que os futuros usuários fossem aos poucos avaliando os protótipos criados, evitando surpresas desagradáveis ao final de todo o processo. Também, que a utilização do framework Rails para o desenvolvimento da aplicação permitiu agilidade na codificação, uma vez que ele é próprio para o desenvolvimento de sites orientados a banco de dados, que é exatamente o caso de uma aplicação de gerenciamento de estoques.

Através do feedback contínuo dos colaboradores do LPTox e da avaliação do protótipo atual, que constatou como 70% dos casos de uso já concluídos, conclui-se que a aplicação já pode trazer benefícios reais para o laboratório. Também, que ainda

necessita de complementação, principalmente a implementação do requisito de filtragem. Com isso, avalia-se que o projeto já cumpriu com grande parte de seus objetivos traçados, e que se pretende finalizar os que ainda faltam.

Referências

BIGG, Ryan et al. Getting Started with Rails. Disponível em: <http://guides.rubyonrails.org/getting_started.html#what-is-rails-questionmark>. Acesso em: 14 out. 2016.

FIRPO, Fernando. Análise de Requisitos: Processos de Obtenção de Requisitos. 2011. Disponível em: <http://analiserequisitos.blogspot.com.br/2011/07/processos-de-obtencao-de-requisitos.html>. Acesso em: 2 jul. 2016

FLANAGAN, David; MATSUMOTO, Yukihiro. The Ruby Programming Language. Df: O'reilly, 2008. 444 p. Disponível em: <http://haris-krajina.rhcloud.com/wp-content/uploads/2013/01/baba.pdf>. Acesso em: 05 out. 2016.

KULKARNI, Jayant et al. Quartzy. Disponível em: <https://www.quartzy.com/about>. Acesso em: 18 out. 2016.

RUBY, Sam; THOMAS, Dave; HANSSON, David Heinemeier. Agile Web Development with Rails 4. 4. ed. Dalas: Rails, 2013. Disponível em: <https://docente.ifrn.edu.br/fellipealeixo/disciplinas/tads-2012/desenvolvimento-de-sistemas-web/book/livro-texto>. Acesso em: 10 ago. 2016.

RUBY. About Ruby. Disponível em: <https://www.ruby-lang.org/en/about/>. Acesso em: 14 out. 2016.

SCRUM. What is a Scrum?. 2016. Disponível em: <https://www.scrum.org/Resources/What-is-Scrum>. Acesso em: 24 ago. 2016.

VERHEYEN, Gunther. Scrum: Framework, not methodology. 2013. Disponível em: <Scrum: Framework, not methodology>. Acesso em: 22 ago. 2016.